1. Inspiração biológica e Motivação

Na natureza, os insetos utilizam os mais diversos mecanismos de defesa, ataque e reprodução. Notadamente, os vaga-lumes se destacam pela capacidade de emitir luz para afastar predadores, atrair presas e atrair parceiros do sexo oposto. Os machos se iluminam para atrair as fêmeas, que tenderão a escolher os mais brilhantes. Por isso, cada macho concorre com os demais tentando ser o mais brilhante e conseqüentemente atrair as fêmeas. O mecanismo utilizado pelos vaga-lumes para gerar a sua bioluminescência é de alto rendimento, da energia que produzem, cerca de 90% se transforma em luminosidade e apenas 10% é dissipado em calor. Seus órgãos bioluminescentes estão localizados na parte inferior dos segmentos abdominais, nos quais ocorrem reações químicas que geram luminescência no inseto. A molécula de ATP (Adenosina Trifosfato), armazenadora de energia, provoca uma reação entre o oxigênio e uma substância chamada luciferina, juntamente com a ação da enzima luciferase, produzindo a molécula de oxiluciferina que encontra-se no estado excitado, ou seja, houve acúmulo de energia na molécula. Então a energia da oxiluciferina é liberada em forma de luz.

Já existem vários algoritmos de otimização e busca inspirados em colônias de animais como: bando de pássaros[1], cardume de peixes[2], enxame de abelhas[3], colônia de formigas[4], entre outros. No entanto, a maioria desses algoritmos visam localizar apenas um ponto ótimo em uma função, o ótimo global, mesmo que a função seja multimodal, ou seja, possui mais de um ponto ótimo. Há duas classes de problemas relacionados à otimização de funções multimodais. A primeira, tradicional, é focada no desenvolvimento de algoritmos capazes de encontrar os ótimos globais, de uma dada função multimodal, excluindo os ótimos locais. Enquanto o segunda classe é focada na localização tanto dos ótimos globais quanto os ótimos locais. Conhecer os múltiplos ótimos de uma função multimodal traz diversas vantagens, tais como prover uma idéia do comportamento da função permitindo a escolha de soluções alternativas quando as ótimas são inviáveis devido a restrições do problema, ou ainda em caso das restrições serem dinâmicas que podem tornar soluções anteriormente viáveis em inviáveis de serem implementadas. Problemas de busca e otimização multimodais são desafiadores para os algoritmos de otimização, pois existem diversos atratores no espaço de busca, com isso o algoritmo ter capacidade de encontrar o ótimo global nessas condições. Se tratando de localizar os múltiplos ótimos, globais e locais, os problemas são ainda mais complexos.

Os cientistas K. N. Krishnanand e D. Ghose desenvolveram em 2005 a primeira versão de um algoritmo de otimização multimodal baseado no comportamento dos vaga-lumes, denominado Otimização por Enxame de Vaga-lumes (GSO - *Glowworm Swarm Optimization*). O principal aspecto dos vaga-lumes utilizado nesse algoritmo é a capacidade deles atraírem os outros através da luminescência. Da forma que o algoritmo foi modelado os agentes (vaga-lumes) têm a capacidade de se agruparem em sub-enxames, os quais se concentram nos

diversos atratores do espaço de busca possibilitando a localização de múltiplas soluções ótimas em um dado problema.

2. Visão Geral

O GSO é uma técnica capaz de localizar simultaneamente múltiplos ótimos em funções multimodais. O algoritmo apresenta algumas semelhanças com o PSO, no entanto possui diferenças significantes que o caracteriza como um novo algoritmo e não apenas uma extensão do PSO. As diferenças estão presentes nas equações envolvidas no processo e, principalmente, na capacidade de criação dinâmica de sub-enxames. No GSO os agentes são representados como vaga-lumes, que carregam uma quantidade de luminescência chamada luciferina[1]. Cada vaga-lume utiliza um valor artificial equivalente à luciferina para divulgar para seus vizinhos o *fitness* da sua localização atual no espaço de busca. A avaliação do *fitness* depende da função objetivo, a qual está ligada diretamente ao problema que será abordado pela técnica. Cada vaga-lumes possui um sensor radial limitado que o capacita a criar sua vizinhança dinamicamente. A partir de sua vizinhança o vaga-lume pode decidir a direção do seu movimento. Ele decide, com base em um mecanismo probabilístico, qual o vizinho mais brilhante que deverá seguir, isto é, os vaga-lumes são atraídos pelos seus vizinhos que têm maior brilho, valor de luciferina. Esses movimentos são baseados apenas nas informações locais e das interações entre os vaga-lumes de cada vizinhança, isso possibilita que o enxame se divida formando subgrupos que convergirão para os múltiplos ótimos de uma dada função multimodal.

[1] Como visto na seção 1 existem outras substâncias responsáveis por gerar luminescências nos vaga-lumes, mas para simplificação a quantidade de luminescência será chamada apenas de luciferina.

3. Funcionamento

No GSO, inicialmente, uma população de n agentes (vaga-lumes), identificados pelo conjunto $\{i : i = 1,2,...,n\}$, é distribuída aleatoriamente ao longo do espaço de busca m-dimensional. As posições no tempo discreto t são representadas por $x_i(t)$ e o conjunto de posições iniciais é dado por $\{x_i(0) : x_i \in \mathbb{R}^m, i = 1, 2, ..., n\}$. É importante que os agentes sejam distribuídos da maneira mais uniforme possível, pois essa técnica não possui um mecanismo eficiente para gerar diversidade na busca, isso a torna mais dependente da inicialização das posições dos vaga-lumes. Cada agente no enxame decide sua direção de movimento pela força do sinal capturado dos seus vizinhos. Isso é algo similar a incandescência induzida da luciferina de um vaga-lume, usada para atrair parceiros e presas. Quanto mais brilhante a luz, maior é a atração. Os vaga-lumes são incorporados nesta técnica com outros mecanismos de comportamento (não presente na realidade) que permite a eles interagirem seletivamente com seus vizinhos e decidir seus movimentos em cada iteração. Na natureza, o brilho do vaga-lume é reduzido com o aumento da distância para a fonte emissora. No GSO não é usado esse tipo de decremento.

Inicialmente todos os vaga-lumes contêm uma quantidade igual de luciferina l_0. Em seguida eles avaliam a função objetivo na sua posição atual e produzem, equivalentemente, um valor de luciferina l_i que é divulgado dentro da sua vizinhança. Cada vaga-lume considera somente aquelas luciferinas que são emitidas pelos seus vizinhos; o conjunto de vizinhos de um vaga-lume consiste naqueles que têm um valor relativo de luciferina maior e que estão localizados dentro de um domínio de decisão adaptativo r_d^i (também podendo dinâmico ser chamado de raio de vizinhança) cujos limites são definidos por um sensoriamento radial r_s ($0 < r_d^i \leq r_s$), onde r_d^i é variável e r_s é fixo. Em outras palavras, um vaga-lume i considera outro vaga-lume j como seu vizinho se j está dentro do raio de vizinhança de i e a quantidade de luciferina de j é maior do que a de i. Para melhor entendimento da formação da vizinhança considere a **Figura 1**. O agente i está dentro do raio máximo de sensoriamento de j e k, no entanto, mesmo que a quantidade de luciferina de i seja maior do que a de j e k, somente j poderá usar as informação de i e mover-se em sua direção, pois i encontra-se apenas no interior do raio de decisão adaptativo r_d^j de j, ou seja, $d(i,j) < r_d^j < r_s$. A **Figura 2**, mostra o grafo direto de vizinhança baseado na quantidade relativa de luciferina nos agentes. Nela os agentes estão ranqueados de acordo com o valor de luciferina, o vaga-lume "a" é o que possui o maior valor. Nesse exemplo, o vaga-lume "f" pode seguir o "e" ou o "c", o "e" pode seguir o "d" ou o "c" e assim por diante.

Estabelecida a vizinhança o vaga-lume seleciona um vizinho com uma probabilidade $p_{ij}(t)$ e se move na direção dele. Esses movimentos, baseados somente na informação local, permitem que os vaga-lumes se dividam em subgrupos disjuntos, realizem um comportamento simultâneo direcionado e tendam a localizar os múltiplos pontos ótimos da função objetivo dada.

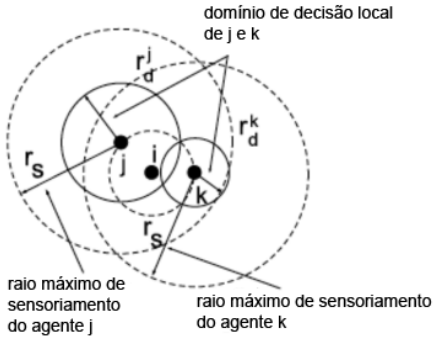

Figura 1: Raio máximo de sensoriamento e raio dinâmico de vizinhança

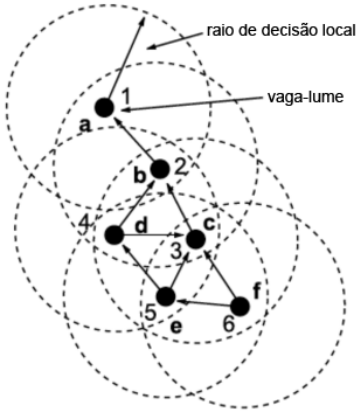

Figura 2: Grafo direto da vizinhança dos vaga-lumes

4. Fases do algoritmo

O GSO possui três fases, as quais são executadas a cada iteração do algoritmo. As fases são:

- Atualização de luciferina
- Movimentação dos vaga-lumes
- Atualização do raio de vizinhança

Em seguida cada fase será detalhada.

4.1 Atualização de luciferina

Esta fase está diretamente ligada ao valor da função objetivo na posição atual do vaga-lume. Para atualizar sua quantidade de luciferina, cada vaga-lume, adiciona uma quantidade proporcional ao *fitness* obtido na sua posição atual ao valor de luciferina que já possuía anteriormente. Para promover uma busca mais suave e aumentar a capacidade de exploração do algoritmo é feita uma subtração do valor de luciferina para simular o decaimento da luciferina ao longo do tempo. Esse fator de decaimento ρ $(0 < \rho < 1)$ evita que algum vaga-lume atinja uma quantidade

muito elevada de luciferina e conseqüentemente os vaga-lumes próximos a ele seriam atraídos com muita intensidade, dificultando a formação de outras vizinhanças. Além disso, os vaga-lumes que não foram tão bem sucedidos nas primeiras iterações da busca dificilmente conseguiriam atrair outros vaga-lumes, pois seus respectivos valores de luciferina provavelmente estariam bem abaixo dos valores de luciferina dos mais bem sucedidos. Em suma, esse fator reduz a possibilidade de ocorrer estagnação na busca. A regra de atualização da luciferina é dada pela seguinte equação:

$$l_i(t+1) = (1-\rho)l_i(t) + \gamma J(x_i(t+1)), \tag{1}$$

onde $l_i(t)$ é o valor de luciferina associado ao vaga-lume i no tempo t, γ é a taxa de incremento da luciferina e $J(x_i(t))$ é o valor da função objetivo no local do vaga-lume i no tempo t.

Da maneira que foi proposto o GSO, os parâmetros ρ e γ são fixos ao longo das iterações. Entretanto, pode ser interessante incorporar tais parâmetros de forma auto-adaptativa no processo de busca, com isso eliminando esses parâmetros. Outra abordagem seria variar esses parâmetros proporcionalmente ao número de iterações.

4.2 Movimentação dos vaga-lumes

Durante a fase de movimentação cada vaga-lume decide, através de um mecanismo probabilístico, se mover na direção de algum vizinho que possuir o valor de luciferina maior que o seu, ou seja, os vaga-lumes são atraídos pelos vizinhos mais brilhantes. Além disso, os movimentos são efetuados com base apenas nas informações locais. Por exemplo, na **Figura 2** o vaga-lume "f" é o que possui menor valor de luciferina dentre os seis, mas ele só pode se mover na direção de "e" ou de "c", pois são os únicos que estão no seu raio de vizinhança. Para cada vaga-lume i, a probabilidade dele mover-se na direção de um vizinho j é dada por:

$$p_{ij}(t) = \frac{l_j(t) - l_i(t)}{\sum_{k \in N_i(t)} l_k(t) - l_i(t)}, \tag{2}$$

onde $j \in N_i(t)$, $N_i(t)$ é o conjunto de vizinhos do vaga-lume i no instante de tempo t e é dado por $N_i(t) = \{j : d_{ij}(t) < r_d^i(t); l_i(t) < l_j(t)\}$, no qual $d_{ij}(t)$ é a distância euclidiana entre os vaga-lumes i e j no tempo t e $r_d^i(t)$, como já descrito, é o raio de vizinhança associado ao vaga-lume i no tempo t. A equação (2) simula uma roleta (comumente usada nos Algoritmo Genéticos [6]), pela qual os vaga-lumes tendem a seguir os vizinhos mais brilhantes, pois $p_{ij}(t)$ é proporcional a $l_j(t) - l_i(t)$. O fato dos vaga-lumes escolherem, probabilisticamente, qual vizinho que irá seguir proporciona uma busca mais refinada na região do espaço de busca em que eles se encontram. Dado que um vaga-lume i escolheu outro vaga-lume $j \in N_i(t)$ a partir de $p_{ij}(t)$, então ele se movimentará de acordo com:

$$x_i(t+1) = x_i(t) + s\left(\frac{x_j(t) - x_i(t)}{\left\|x_j(t) - x_i(t)\right\|}\right), \quad (3)$$

onde $x_i(t) \in \mathbb{R}^m$ é a posição do vaga-lume i, no tempo t, no espaço m-dimensional $\mathbb{R}^m$, s ($s > 0$) é o tamanho do passo na direção do vetor unitário $\left(\frac{x_j(t) - x_i(t)}{\left\|x_j(t) - x_i(t)\right\|}\right)$, lembrando que || . || representa a norma euclidiana do vetor.

Da forma que foi proposto o GSO, o parâmetro s é fixo ao longo das iterações. Há um motivo plausível para fixar esse parâmetro, que é o *leapfrog behavior*. Esse comportamento será detalhado mais adiante, mas ele basicamente gera um comportamento similar à uma busca por *Hill Climbing*. Se o parâmetro s for muito elevado os vaga-lumes vão mudar suas posições de forma drástica, acarretando em uma busca pouco refinada. Por outro lado, se s for muito pequeno os vaga-lumes farão movimentos muito pequenos o que deve levar a uma convergência lenta do algoritmo. Os criadores do GSO, baseados em vários experimentos, sugerem 0,03 como um valor apropriado. No entanto, não há garantia que esse é o valor ótimo para todo tipo de problema.

4.3 Atualização do raio de vizinhança

Como foi visto, cada agente possui um raio de vizinhança dinâmico r_d^i limitado ao intervalo $(0 < r_d^i \leq r_s)$. Nessa fase ocorre o aumento ou a redução desse raio de vizinhança de acordo com a quantidade de vizinhos. Existem algumas justificativas para o raio de vizinhança ser variável. Primeiro, se o raio de vizinhança de cada agente fosse grande demais, ao ponto de cobrir todo espaço de busca, todos os vaga-lumes iriam convergir para o ótimo global ignorando os ótimos locais, pois seria formada apenas uma grande vizinhança, já que todos se "enxergariam", contendo todos os vaga-lumes, que tenderiam a seguir o mais brilhante que provavelmente estaria próximo ao ótimo global. Esse efeito levaria à perda da capacidade de formação de sub-enxames, pois as interações entre os agentes não seriam apenas locais e sim global. Segundo, se o valor de r_d^i for muito pequeno, seria mais difícil ocorrer a formação de vizinhanças, além de aumentar a chance de alguns vaga-lumes ficarem isolados e não se moverem por não terem vizinhos para seguir. E ainda, uma terceiro justificativa, é que a priori não se tem informações sobre a função objetivo no caso de problemas reais (por exemplo, distância entre os picos e a quantidade de picos), sendo assim é difícil encontrar um valor de raio de vizinhança que funcione bem para qualquer tipo de função objetivo. Por exemplo, um valor de raio de vizinhança poderia funcionar melhor em funções objetivo que têm distância entre picos maior que r_s do que naquelas menor que r_s. Por esses motivos o GSO utiliza o raio de vizinhança de forma adaptativa para possibilitar a detecção de múltiplos picos em uma função multimodal.

O raio de vizinhança inicial para todos os vaga-lumes é dado pelo parâmetro r_0, ou seja, $r_d^i(0) = r_0 \, \forall i$. Para realizar a atualização do raio de vizinhança de forma adaptativa o GSO utiliza a seguinte regra:

$$r_d^i(t+1) = \min\{r_s, \max\{0, r_d^i(t) + \beta(n_t - |N_i(t)|)\}\}, \quad (4)$$

onde β é a taxa de mudança do raio de vizinhança e n_t é a quantidade máxima de vizinhos.

Para facilitar o entendimento será mostrado um exemplo da variação das vizinhanças e atualização do raio de vizinhança. Considere que os vaga-lumes estarão estáticos e observe a variação do raio de vizinhança do vaga-lume que está na posição (0,0) da **Figura 3** de acordo com a equação (4). Nesse exemplo n_t=3 e r_s =1. Inicialmente, o vaga-lume posicionado em (0,0) está isolado. Ao longo das iterações ele vai aumentando seu raio de vizinhança até encontrar a quantidade máxima de vizinhos n_t ou o raio de vizinhança atingir o valor de r_s. O comportamento do tamanho o raio de vizinhança, bem como a quantidade de vizinhos, são mostrados na **Figura 4**. Note que, quanto menor a quantidade de vizinhos, mais acelerado é o crescimento do raio de vizinhança e quando o vaga-lume atinge o número máximo de vizinhos o seu raio de vizinhança para de aumentar. Esse comportamento é importante para diminuir a possibilidade de isolamento de vaga-lumes, uma vez que quanto menos vizinhos um certo vaga-lume tiver, maior será o aumento do seu raio de vizinhança, justamente para aumentar a chance de encontrar algum vizinho. Os isolamentos podem ocorrer, geralmente, nas regiões mais afastadas dos ótimos.

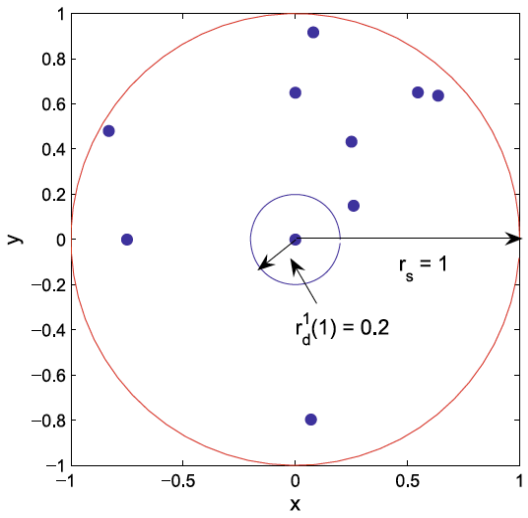

Figura 3: Vaga-lume em isolamento

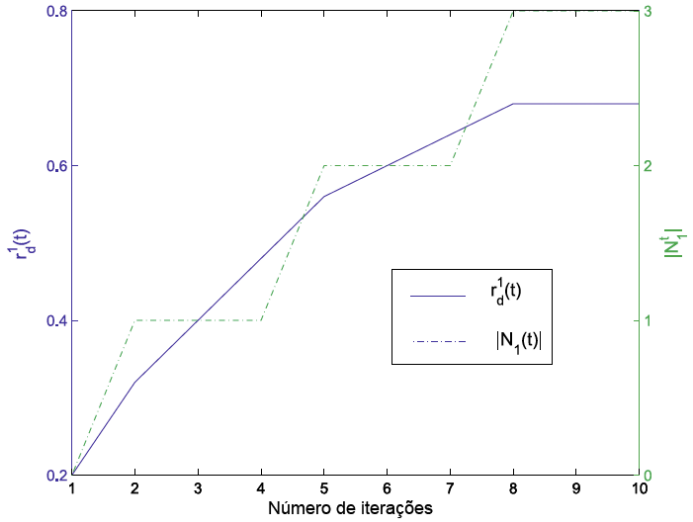

Figura 4: Variação do raio dinâmico de vizinhança e quantidade de vizinhos do vaga-lume 1

Há outro caso, no qual o raio de vizinhança de uma determinado vaga-lume deve ser reduzido. Considere outro exemplo, com as mesmas condições iniciais do exemplo anterior, com a diferença que o vaga-lume da posição (0,0) não está isolado, a **Figura 5** expressa tal situação. Nesse exemplo, como os vaga-lumes devem manter uma vizinhança com no máximo três outros vaga-lumes (n_t=3) então o raio de vizinhança r_d^1 deve ser reduzido até que o vaga-lume de índice "1" não continue com oito vizinhos. A variação do raio de vizinhança e a quantidade de vizinhos de "1" são mostradas na **Figura 6**.

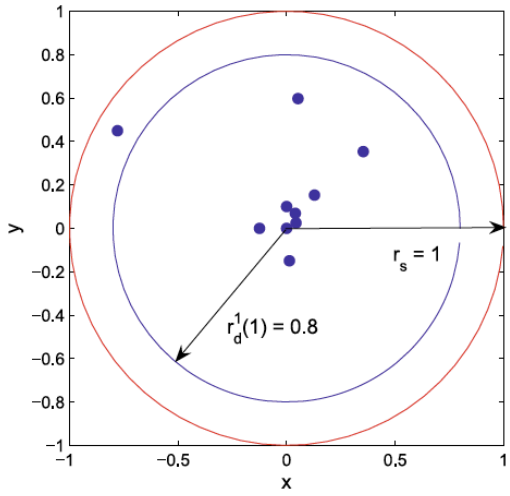

Figura 5: Vaga-lume com valor excessivo de vizinhos

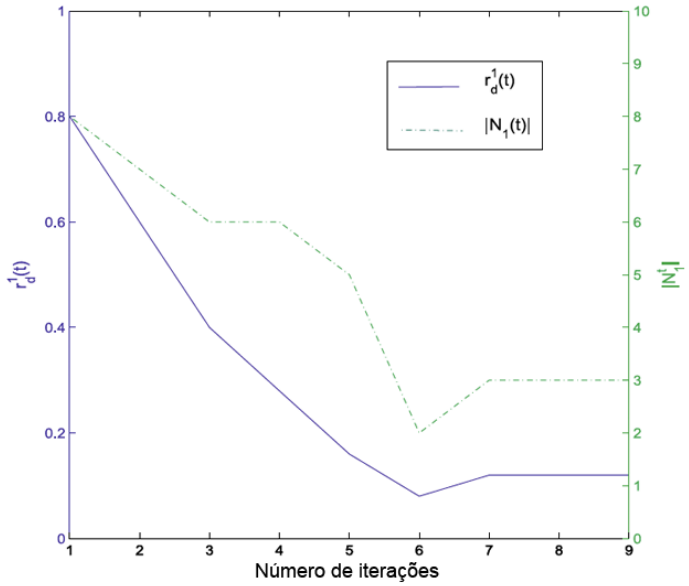

Figura 6: Variação do raio dinâmico de vizinhança e quantidade de vizinhos do vaga-lume 1

5. Operação

A **Figura 7** mostra o pseudo-código do algoritmo GSO.

Inicializa dos parâmetros
Distribui os vaga-lumes aleatoriamente do espaço de busca
Para cada vaga-luma faça
    Setar valor de luciferina como l_0
    Setar valor do raio de vizinhança como r_0 *//* $r_d^i(0) = r_0$
Setar número máximo de iterações
Inicializa o tempo discreto $t = 1$
Enquanto não esgotar a quantidade de iterações
    Para cada vaga-lume faça
        Execute a fase de ***atualização da luciferina***
    Para cada vaga-lume i *faça //****Fase de movimentação***
        Determina a vizinhança de cada vaga-lume // $N_i(t)$
        Para cada vaga-lume $j \in N_i(t)$ *faça*
            Calcule a probabilidade de i *seguir* j
    Atualize a posição de i
        Execute a fase de ***atualização do raio de vizinhança***
    Incrementa o número de iterações //valor de t
Fim enquanto

Figura 7: Pseudo-código do GSO

6. *Leapfrog Behavior*

Pela descrição o algoritmo, pode parece que em alguma iteração o vaga-lume com valor máximo de luciferina poderia ficar estático, pois não teria nenhum outro que tivesse mais luciferina que ele, logo ele não poderia seguir nenhum outro. Se isso acontecesse levaria o algoritmo a uma situação de estagnação, pois todos os vaga-lumes próximos a um ótimo iria convergir para o local do vaga-lume mais próximo do ótimo. Com isso, haveria uma redução da capacidade de exploração em profundidade do algoritmo. Entretanto, essa estagnação não ocorre devido à regra de mudança de movimentação mostrada na equação (3), mais especificamente, pelo fato dos vaga-lumes fazerem movimentos com tamanho fixo, valor do parâmetro s. Quando um a distância entre um vaga-lume i e um vizinho j (com valor maior de luciferina) é menor que s, então i "salta" além da posição de j, se i ficou mais próximo de algum ótimo do que j então i deverá obter mais luciferina e se tornando o líder daquela vizinhança. Nesse momento, j ficará parado e na iteração seguinte deverá se mover em direção a i "saltando" além da posição de i. Esse comportamento irá se repetir, alternando entre movimentos de i e j. A partir disso, emerge um comportamento de busca local, algo similar ao *Hill Cilmbing*. Um grupo vaga-lumes usa esse princípio para melhorar a execução da busca local e eventualmente convergiram para um ponto ótimo. O comportamento descrito pode ser observado na simulação para otimização da função presente na **Figura 8**. A **Figura 9** mostra, na visão do plano XY, a trajetória dos agentes na otimização da função da **Figura 8**. É possível notar que em determinado instante os agentes que estão na região destaque da **Figura 9** realizam seus movimentos bem próximos uns aos outros, todos de dirigindo ao pico presente no ponto (1.28, 0). Isso caracteriza o *leapfrog behavior.*

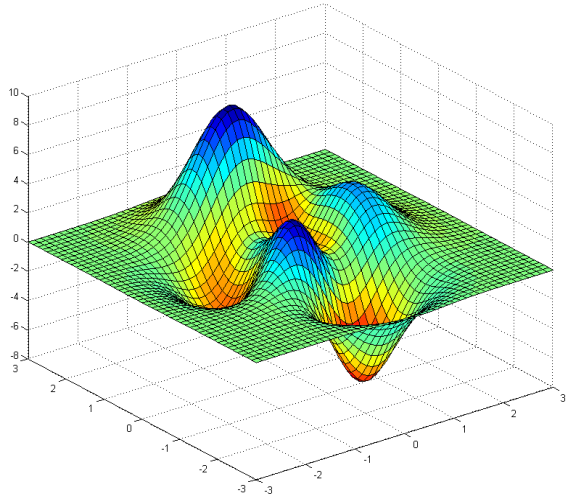

Figura 8: Gráfico da função Picos

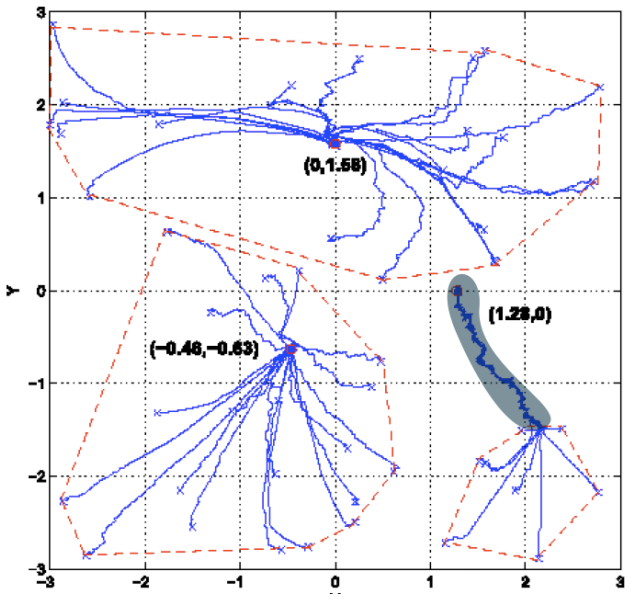

Figura 9: Trajetória dos agentes ao longo das iterações

7. Parâmetros

A quantidade de parâmetros têm um impacto considerável na aplicabilidade do algoritmo. Quanto menos parâmetros um algoritmo tiver melhor ele é para o usuário, uma vez que o ajuste dos parâmetros pode estar diretamente ligado ao problema que é abordado, o que exige um conhecimento prévio do problema. Há algoritmos que possuem parâmetros que não afetam de forma significativa seu desempenho na aplicação em vários problemas. Para chegar a esse tipo de conclusão, é preciso realizar experimentos para analisar a influência da alteração do valor de certo parâmetro no resultado do algoritmo. Em alguns casos há um valor de um parâmetro que é bom para a resolução de um conjunto de problemas, mas não se pode garantir que é o valor ótimo.

O GSO possui nove parâmetros (quantidade relativamente alta). Os criadores da técnica, baseados em vários experimentos, sugerem, para sete parâmetros, valores apropriados para a o conjunto de funções que o algoritmo foi testado. Os parâmetros são (os valores sugeridos estão entre parênteses):

1. n_t (5): número máximo de vizinhos por vaga-lume
2. s (0,03): tamanho do passo
3. l_0 (5): valor inicial de luciferina
4. β (0,08): taxa de mudança do raio de vizinhança
5. ρ (0,4): taxa de decaimento da luciferina
6. γ (0,6): taxa de incremento da luciferina
7. r_0 (r_s): valor inicial do raio de vizinhança
8. r_s: valor máximo do raio de vizinhança
9. n: quantidade de vaga-lume

De acordo com os criadores do GSO, os parâmetros que mais influenciam no desempenho do algoritmo, em termos de número de picos encontrados, são r_s e n.

Referências

[1] KENNEDY, J.; EBERHART, R. Particle swarm optimization, Neural Networks IEEE International Conference, Austrália, v. 4, p. 1942-1948, 1995

[2] BASTOS FILHO, C. J. A.; LIMA NETO, F. B.; SOUSA, M. F. C.; PONTES, M. R.; MADEIRO, S. S. On the Influence of the Swimming Operators in the Fish School Search Algorithm, SMC, 2009

[3] KARABOGA, D., An Idea Based On Honey Bee Swarm for Numerical Optimization, Technical Report-TR06, Erciyes University, Engineering Faculty, Computer Engineering Department 2005

[4] DORIGO, M.; GAMBARDELLA, L.M. Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem, IEEE Transactions on Evolutionary Computation, v. 1, número 1, p. 53-66, 1997

[5] Krishnanand, K.N.; Ghose, D. Glowworm swarm optimization for simultaneous capture of multiple local optima of multimodal functions, Springer, 2008

[6] HOLLAND, J. H. Adaptation in Natural and Artificial Systems, Ann Arbor, MI: The University of Michigan Press, 1975.