TEMA Tend. Mat. Apl. Comput., **7**, No. 1 (2006), 159-168.

O Processo de Otimização Ant System com Redução no Raio de Busca

A. de VICENTE[1], Centro de Ciências Exatas e Tecnológicas, UNIOESTE, Rua Universitária 2069, 85819-110 Cascavel, PR, Brasil.

Resumo. Este artigo apresenta um processo heurístico de otimização para funções reais, que simula o comportamento de uma espécie de formiga à procura de alimentos, processo conhecido como Ant Colony Optimization (ACO) ou Ant System (AS). Esta espécie de formiga é caracterizada por uma estratégia de caça onde cada indivíduo demarca pequenas regiões de busca em torno da colônia, de modo a cobrir todo o espaço em torno da mesma. Além disso, a colônia é mudada periodicamente de lugar à medida que o alimento se torna escasso, o que significa que a pesquisa é feita tanto local quanto globalmente na região que cerca tal colônia. Em experimentos realizados com algumas funções reais clássicas utilizadas para testes o algoritmo mostrou um excelente desempenho fornecendo sempre uma solução idêntica ou melhor do que a solução obtida por outros algoritmos similares.
Palavras-chave: Otimização, Ant System, Programação não-linear

1. Introdução

Apesar de todas as pesquisas já realizadas, encontrar um método determinístico que determine o ótimo global de uma função é uma tarefa que ainda está sem solução. Por causa disso, as pesquisas têm se voltado para os métodos não-determinísticos (ver [3]), que procuram o ponto extremo de uma função geralmente tentando imitar um fenômeno da natureza. O funcionamento destes métodos geralmente permite fugir de ótimos locais, o que dá grandes chances de que seja obtida uma solução próxima do ótimo global. Esses métodos são dotados de regras especiais, que advêm do fato de imitarem fenômenos naturais, que fazem com que eles não sejam um simples processo aleatório, mas sim uma forma de pesquisa simples e inteligente na busca de soluções para os mais variados tipos de problema. Neste trabalho será apresentado um algoritmo para o processo Ant System, conforme será descrito nos itens a seguir.

2. Descrição do Processo AS para Funções Reais

Originalmente o AS foi desenvolvido para resolver o TSP (Problema do Caixeiro Viajante) e o algoritmo básico foi construído para este fim (ver [2]). Da forma como

[1]amarildo@unioeste.br.

foi apresentado originalmente o AS se aplicava apenas a problemas que permitiam uma representação por meio de grafos, o que não é o caso de um problema de otimização de uma função real num domínio contínuo. Para este tipo de problema foi desenvolvido um trabalho (ver [1]) onde se segue a estratégia de um tipo especial de formiga (*Pachycondyla apicalis*) na busca de alimento. O interesse sobre estas formigas para otimização se deve ao fato de que elas usam mecanismos simples bem determinados para capturarem suas presas, muito parecidos com um processo de busca de um ponto de máximo para uma função em seu domínio. Este processo de caça realizado pelas formigas é feito tanto de um ponto de vista local quanto global.

Sob um ponto de vista global a busca é feita da seguinte maneira: Partindo de seu ninho elas cobrem globalmente uma dada superfície dividindo-a em pequenas regiões de caça individuais. Estas regiões de caça são criadas uniformemente em torno do ninho e têm seus centros localizados a uma distância máxima (em média) de aproximadamente 10 metros do mesmo, com uma amplitude de 2.5 metros aproximadamente. Com isso as formigas cobrem com um mosaico de pequenas regiões uma grande superfície ao redor de sua moradia. Periodicamente se observam trocas de local do ninho. Estas trocas podem ocorrer, por exemplo, por dilapidação da atual moradia ou por escassez de alimento naquele local. A mudança do ninho é um processo complexo que usa formigas especializadas, tanto na busca de um novo local como em mecanismos de recrutamento para fazer toda a colônia mudar para o novo ninho.

Localmente a estratégia para a procura de alimentos é a seguinte: Inicialmente uma formiga escolhe aleatoriamente uma região de caça em torno do ninho. Após obter um sucesso ela memoriza a região onde a presa foi capturada, a fim de retornar para esta mesma região. O retorno é feito com o auxílio de uma trilha que ela mesma produz. Esta trilha não é formada por produtos químicos mas sim por marcas deixadas no solo. Quando uma presa é capturada, a próxima busca a ser realizada pela formiga inicia sempre na respectiva região de captura. Quando esta região empobrece e a formiga demora para obter outra presa, ela tem tendência a explorar outras direções. Futuramente tais formigas podem voltar para uma região previamente explorada, o que realça sua habilidade em memorizar regiões.

Sempre que uma formiga captura uma presa ela a leva para o ninho onde a mesma é depositada. Outro fato observado é que, à medida que envelhecem, as formigas vão ficando mais "corajosas", incrementando seu raio de ação em torno do ninho.

3. Modelo para uma Colônia

Para simular o comportamento dessas formigas torna-se necessária a criação de um modelo para a colônia. Para isso, considere-se uma população de n formigas f_1, f_2,..., f_n, localizadas num domínio de pesquisa contínuo D, e uma função g: D $\rightarrow \mathbb{R}$ a ser maximizada. Cada ponto c $\in$ D é considerado como uma solução válida para o problema. Considere-se ainda os seguintes operadores:

1. OP1, que gera pontos aleatórios em D uniformemente distribuídos.
2. OP2, que gera pontos aleatórios em D numa vizinhança de um ponto c dado.

Para o segundo operador, considera-se que a vizinhança centrada em c seja uma bola fechada com raio r. Este raio estabelece a distância máxima de exploração em torno de c.

Sob o ponto de vista global, o procedimento de exploração consiste em estabelecer o local do ninho N em D e realizar explorações em torno de tal ninho. No início N é posto em D de forma randômica, por meio do operador OP1. A partir daí, a cada M simulações de busca de cada uma das n formigas (avaliação de g em M vezes n pontos) o ninho é movido para o melhor ponto encontrado desde a última mudança.

Sob o ponto de vista local, inicialmente e a cada vez que o ninho for movido, cada formiga f_i deixa o ninho para criar R regiões de caça em sua memória, de acordo com uma distribuição uniforme, centradas em torno do ninho. Para criar estas regiões f_i utiliza OP2 para gerar os centros c_1, c_2,..., c_R das regiões, a uma distância máxima r_i do ninho N (Figura 1-a). Cada formiga f_i procura por uma presa em sua região de caça realizando a exploração de uma região selecionada (Figura 1-b).

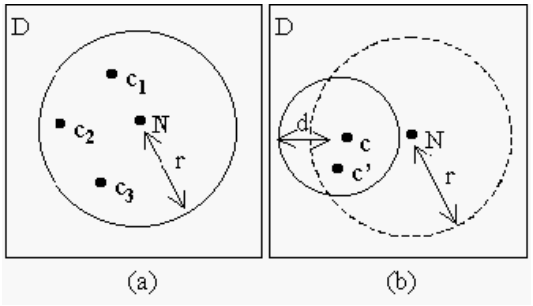

Figura 1: Colônia de formigas num domínio contínuo D
(a) Centros de regiões de caça situados a uma distância máxima r do ninho N.
(b) Ponto de exploração c' situado a uma distância máxima d do centro c de sua região.

No início, quando a região de interesse não é conhecida, f_i seleciona aleatoriamente uma das R regiões e vai para lá. Seja c o centro desta região. Então f_i realiza uma busca local situada em um ponto c', em torno de c, através do operador OP2, com uma distância máxima d_i de c. Uma presa é obtida por f_i se o local de exploração c' conduzir a um valor melhor para g do que aquele obtido em c, isto é, se g(c') > g(c). Portanto, uma melhoria no valor de g significa a realização de uma captura. Além disso, cada vez que uma exploração local é bem sucedida f_i memoriza este sucesso e retorna precisamente a esta região na sua próxima busca. Se a exploração local de uma região de caça não for bem sucedida, então, para a próxima exploração, f_i escolhe aleatoriamente uma região entre as R regiões de sua memória. Se uma região tiver sido explorada sucessivamente por mais de K_i vezes, sem a captura de nenhuma presa, então esta é esquecida definitivamente e, na próxima exploração, é substituída pela criação de uma nova região em torno do ninho.

Esta nova região é criada usando OP2 com o centro situado à mesma distância máxima do ninho já citada, r_i. K_i representa um parâmetro de persistência local. É o número máximo de vezes que a formiga f_i realiza busca seguidamente em uma mesma região sem obter sucesso. Finalmente, em cada nova disposição do ninho as formigas perdem as trilhas que conduziam às suas regiões de caça. As regiões são portanto restabelecidas em suas memórias e, com isso, espera-se uma contribuição para evitar mínimos locais. A Figura 2 ilustra o comportamento de uma formiga genérica.

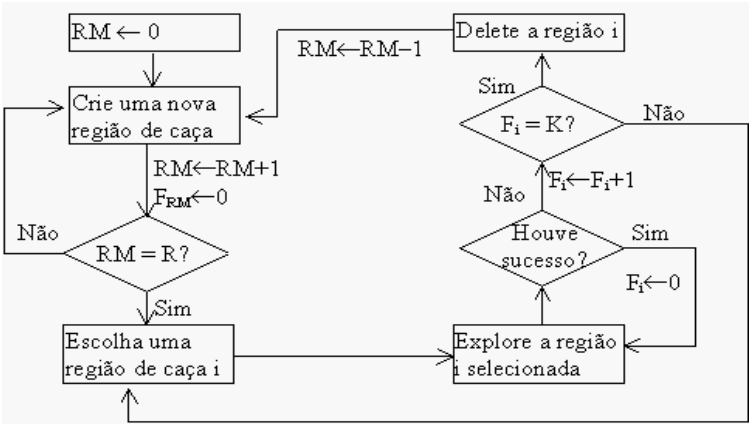

Figura 2: Esquema de exploração para uma formiga genérica
RM = número de regiões memorizadas; F_i = número de fracassos observados na região i; K = número máximo de fracassos admitidos para uma região; P = número de regiões que a formiga é capaz de memorizar.

No processo original o raio máximo de busca das formigas além da colônia é fixo e, conforme mencionado anteriormente, tem em média 10 metros. Porém, ao ser encontrada uma fonte de alimento farta, as formigas tendem a se concentrar neste ponto mudando a colônia para o mesmo. A partir daí o raio de ação de tais formigas é reduzido. Matematicamente isto significa que o ponto ótimo alcançado vai ficando cada vez mais preciso à medida que o raio de ação vai sendo diminuído. A implementação deste fato é a principal diferença entre o algoritmo original, onde esta questão não é levada em consideração, e o algoritmo que está sendo proposto neste trabalho. Por este motivo o algoritmo a ser apresentado na próxima seção será chamado de Algoritmo Ant System com Decréscimo de Raio (ASDR).

4. Algoritmo Ant System com Decréscimo de Raio (ASDR)

Dados iniciais
- NFm: Número de formigas da colônia (ninho).
- NMaxReg: Número máximo de regiões a serem memorizadas por formiga.
- NFracMax: Número de fracassos tolerados por formiga.
- MaxIter: Número máximo de iterações a serem realizadas entre cada mudança da colônia. Uma iteração representa uma exploração para cada formiga da colônia.
- MaxMudCol: Número máximo de mudanças da colônia.
- RMaxFor: Raio máximo a ser explorado em torno do ninho.
- α: fator de redução dos raios de busca das formigas ($0 < \alpha < 1$)
- P: Taxa de probabilidade para recrutamento ($0 \leq P \leq 1$).
- Função f a ser minimizada;

Variáveis
- RFm_i: Raio de ação da formiga i. É a distância máxima percorrida pela formiga i além da colônia.
- $RReg_j$: Raio da região j.
- $Centro_j$: Centro da região j.
- $UltRegExp_i$: Última região explorada pelo formiga i.
- $ResBc_j$: Resultado da última busca na região j (1 para sucesso ou 0 para fracasso).
- $ValF_j$: Valor da função f no centro da região j.
- $NFrac_j$: Número de fracassos sucessivos observados na região j.
- $NRegMem_i$: Número de regiões memorizadas pela formiga i em um dado instante.
- *Iter*: Número de iterações realizadas até um dado instante, entre cada mudança do ninho (cada iteração compreende uma exploração por formiga, o que representa NFm avaliações da função f).
- *MudCol*: Total de mudanças da colônia até um dado instante.
- *MinValF*: Mínimo valor da função.
- *NOrdReg*: Número de ordem de uma região gerada.
- $NOrdReg_{j,i}$: Número de ordem da região j memorizada pela formiga i.

Rotina ExploraRegião(CodReg, CodFor);
CodReg = Código (número) da região e CodFor = Código (número) de uma formiga
Início
    j:=CodReg; i:=CodFor;
    Gere um ponto X na região j e calcule $f(X)$;
    se $f(X) < ValF_j$ então
        $ValF_j := f(X)$;
        $Centro_j :=$ X;
        $Nfrac_j := 0$;
        Faça um recrutamento para a formiga i com taxa de probabilidade P;
    fim {se};

```
    senão
        Nfrac_j := Nfrac_j + 1;
        ResBc_j := 0;
            se Nfrac_j > NFracMax então
            Gere outro centro e outro raio para a região j;
    fim {senão};
fim;

Rotina SimulaFormiga;
Início
    Para i := 1 até NFm faça
        Se NRegMem_i < NMaxReg então
            NordReg := NordReg + 1;
            Gere a região NOrdReg com centro a uma distância máxima
            RFm_i do ninho e com amplitude máxima RFm_i/2;
            ValF_j := f(Centro_j);
            NRegMem_i := NRegMem_i + 1;
            j := NRegMem_i;
            UltRegExp_i := NOrdReg;
            NOrdReg_{j,i} := NOrdReg;
        fim {se}
        senão
            se a última região explorada pela formiga i teve êxito então
                ExploraRegião(UltRegExp_i, i);
            senão
                Selecione aleatoriamente uma região S da memória da formiga i;
                ExploraRegião(S, i);
            fim {senão};
        fim {senão}
    fim {i};
fim;

Início do programa
    MudNin := 0;
    NRegMem_i := 0 para todo i;
    Gere aleatoriamente o ninho N e os raios RFm_i ≤ RMaxFor;
    repita
        MinValF := f(N);
        NregGer := 0; Iter := 0;
        repita
            Iter := Iter + 1;
            SimulaFormiga;
        até que Iter = MaxIter;
        Procure o melhor ponto X encontrado até o momento;
        Se X ≠ N então
            Atualize o valor MinValF para f(X);
```

N := X;
fim se
senão
Se RmaxFor > precisão, então reduza os raios RFm_i e RMaxFor em α %;
$NRegMem_i := 0$ para todo i;
$MudCol := MudCol + 1$;
Até que $MudCol$ = MaxMudCol;
Saída: O ponto de mínimo é o ninho N e o menor valor é $MinValF$ ($f(N)$).
fim.

Nota: Para resolver um problema de maximização com este algoritmo basta trocar o sinal da função em questão.

5. Validação do Algoritmo ASDR e Discussões

Para testar a eficiência do algoritmo proposto foram avaliadas diversas funções e os resultados obtidos foram comparados com os resultados publicados no trabalho de Wodrich (ver [4]). Este trabalho mostra os valores máximos obtidos para diversas funções através dos seguintes métodos: Genetic Algoritms (GA), Multiple-Restart Stochastic Hill-climbing (MSH), Discrete Ant Colony Optimization (DACO), Continuous Ant Colony Optimization (CACO) e Population-Based Incremental Learning (PBIL).

5.1. Função 1 (Dimensão 2)

$$f_1(x,y) = 3905.93 - 10(x^2 - y)^2 - (1-x)^2,\ -2.048 \leq x, y \leq 2.048.$$

Esta função tem como ponto de máximo P = (1, 1), sendo $f_1(1, 1) = 3905.93$. Este resultado foi encontrado por todos os algoritmos testados, conforme pode ser visto na Tabela 1.

Método	Máximo encontrado
DACO	3905.93
GA	3905.93
PBIL	3905.93
MSH	3905.93
ASDR	3905.93

Tabela 1: Resultados obtidos para a função f_1.

5.2. Função 2 (Dimensão 10)

$$f_2(X) = \frac{1}{0.1 + (\sum_{i=1}^{10} \frac{{x_i}^2}{4000} - \prod_{i=1}^{10} cos(\frac{x_i}{\sqrt{i}}) + 1)}, \; -512 \leq x_i \leq 512, \; i = 1, 2, ..., 10.$$

Esta função possui máximo global 10 na origem. Os resultados obtidos estão apresentados na Tabela 2.

Método	Máximo encontrado
DACO	10.00
MSH	10.50
PBIL	5.49
GA	3.52
CACO	10.00
ASDR	10.00

Tabela 2: Resultados obtidos para a função f_2.

Conforme pode ser visto neste quadro, os Algoritmos MSH, PBIL e GA não foram capazes de encontrar o ponto ótimo.

5.3. Função 3 (Dimensão 2)

$f_3(x, y) = -(x-2)^2 - (y-1)^2$
sujeita às restrições
$x - 2y + 1 = 0, \quad \frac{-x^2}{4} - y^2 + 1 \geq 0$
$-1.82 \leq x \leq 0.82, \quad 0.41 \leq y \leq 0.92.$

Neste caso tem-se um problema restrito. Para utilizar o ASDR as restrições foram incorporada à função f através de uma função de penalidade. Com quatro casas decimais este problema possui máximo -1.3935. Os resultados obtidos estão apresentados na Tabela 3.

Método	Máximo encontrado
DACO	-1.3773
GA	-1.4339
CACO	-1.4609
ASDR	-1.3935

Tabela 3: Resultados obtidos para a função f_3.

Para este problema apenas o ASDR foi capaz de localizar o ponto de máximo corretamente. O valor obtido pelo DACO é maior porém é um valor inviável, ou seja, o ponto que o gerou está fora da região viável.

5.4. Função 4 (Dimensão 100)

$$f_4(X) = \frac{1}{10^{-5} + |x_i| + \sum_{i=2}^{100} |x_i - x_{i-1}|}, -2.56 \leq x_i \leq 2.56, i = 1..100.$$

Para esta função o máximo global, que é de 100000, ocorre na origem. A grande dificuldade neste problema é que função é mal condicionada na origem, onde ocorre o ponto de máximo. Além disso, neste ponto as derivadas são descontínuas, fato que dificulta a aplicação de métodos onde as derivadas são necessárias. O resultados obtidos para esta função estão a presentados na Tabela 4.

Método	Máximo encontrado
DACO	0.0684
MSH	0.0121
CACO	0.0126
PBIL	0.0212
GA	0.0196
ASDR	0.1297

Tabela 4: Resultados obtidos para a função f_4.

Observa-se que para este problema o ASDR produziu um resultado maior do que os demais algoritmos, apresentando uma melhoria de quase 90% sobre o maior resultado antes encontrado, que é de 0.0684 para o DACO.

5.5. Função 5 (Dimensão 100)

$$f_5(X) = \frac{1}{10^{-5} + |x_1| + \sum_{i=2}^{100} |x_i + sen(x_{i-1})|}, -2.56 \leq x_i \leq 2.56, i = 1..10.$$

Esta função também possui máximo global 100000 na origem. Do mesmo modo que no caso anterior, trata-se também de um problema em que a função é mal condicionada no ponto onde ocorre o valor máximo. Além disso as derivadas parciais são todas descontínuas em tal ponto. Os resultados obtidos para esta função estão apresentados na Tabela 5.

Conforme já havia ocorrido no caso anterior o ASDR produziu um resultado muito melhor que os demais algoritmos. Isto comprova que tal algoritmo consegue superar os demais no caso de uma função mal condicionada.

6. Conclusões

Como pode ser observado no item anterior, em todas as situações o ASDR foi tão bom quanto os demais algoritmos testados ou melhor. No pior dos caso ele obteve o mesmo resultado dos demais e na melhor das situações o resultado obtido foi quase 90% melhor do que o melhor valor fornecido pelos demais algoritmos. Pelos

Método	Máximo encontrado
DACO	0.1279
MSH	0.0438
CACO	0.0552
PBIL	0.0400
GA	0.0463
ASDR	0.2372

Tabela 5: Resultados obtidos para a função f_5.

resultados obtidos pode-se notar que o algoritmo ASDR foi mais eficaz que os demais nos problemas em que a função é mal condicionada no ponto ótimo. Assim, pelo bom desempenho observado conclui-se que tal algoritmo é uma boa ferramenta para otimizar funções não lineares.

Abstract. This paper presents an optimization heuristic process for real functions, which simulates a kind of ant searching food, process known by Ant Colony Optimization (ACO) or Ant System (AS). In realized experiments with some classic real functions, usually applied on tests, the algorithm presented had an excellent performance, obtaining always identical or better solution than other similar algorithms.

Referências

[1] M. Monmarché, G. Venturini e M Slimane, On How the Ants Pachycondyla apicalis Are Suggesting a New Search Algorithm, *Laboratoire d'Informatique, Université de Tours, France, Internal Repport 214, E3i* January 1999, 17p.

[2] M. Dorigo e M.G. Luca, Ant Colony for the Traveling Salesman Problem *BioSystem* 1997, 10p.

[3] A. de Vicente, "Um Modelo Matemático para a Estruturação de um Sistema de Produção Agrícola Integrado", Tese de Doutorado, UFSC 1999, 118p.

[4] M. Wodrich, "An Empirical Investigation into an Ant Colony Metaphor for Continuous Function Optimisation", Undergraduate thesis,The Department of Electrical and Electronic Engineering, University of Cape Town, 1996.