DOI: 10.5433/1679-0375.2014v35n1p63

Otimização Heurística por Colônia de formigas com Aplicações em Sistemas de Comunicações

Heuristic Ant Colony Optimization with Applications in Communication Systems

Mateus de P. Marques[1]; Bruno A. Angélico[2]; Taufik Abrão[3]

Resumo

Este trabalho explora a técnica de otimização heurística baseada em colônia de formigas (ACO) aplicada a problemas de otimização complexos, tendo em vista a obtenção de um método de otimização iterativo factível de implementação, aplicável a problemas NP e NP-Completo associados a redes de comunicação sem fio. Neste artigo, discute-se o desempenho e convergência do algoritmo ACO contínuo de valores reais utilizando-se dezenas de funções benchmarks de diferentes dimensões e grau de dificuldade de otimização (diferentes números variáveis e quantidade de ótimos locais). Finalmente, a aplicabilidade do método de otimização heurístico ACO é ilustrada evocando-se o problema do controle de potência em redes CDMA.
Palavras-chave: Otimização por colônia de formigas (ACO). ACO contínuo. Sistemas CDMA. Alocação de Recursos.

Abstract

This work explores the heuristic optimization algorithm based on ant colonies (ACO), deployed on complex optimization problems, aiming to achieve an iterative and feasible method which is able to solve NP and NP-Hard problems related to wireless networks. Furthermore, the convergence and performance of the Ant Colony Optimization algorithm for continuous domains are addressed through dozens of benchmark functions, which in turn, differ on each other regarding the number of dimensions and the difficulty w.r.t. the optimization (number of local optima). Finally, the applicability of the ACO is depicted in an minimum power control problem for CDMA networks.
Key words: Ant Colony Optimization (ACO). ACO continum. CDMA systems.

[1] Mestrando em Engenharia Elétrica, Universidade Estadual de Londrina; mat.pmarques@gmail.com
[2] Docente da Universidade Tecnológica Federal do Paraná; bangelico@utfpr.edu.br
[3] Docente do Departamento de Engenharia Elétrica da Universidade Estadual de Londrina - DEEL-UEL; taufik@uel.br

Introdução

A inteligência de partículas é uma técnica de otimização pertencente ao campo da computação evolucionária com grande interesse de aplicações e franca expansão nos últimos dez anos. Neste contexto, a otimização por colônia de formigas (ACO – Ant Colony Optimization) está baseada na comunicação estigmergética[1] de formigas reais. Inicialmente, a otimização através de colônia de formigas alcançou grande sucesso nas aplicações a problemas de otimização combinatória. Sabe-se que a maioria desses problemas pode ser classificada como NP-Completo, ou seja, atualmente não existe um algoritmo capaz de resolvê-los em tempo polinomial (SCHRIJVER, 2003). O algoritmo ACO para problemas de otimização combinatória foi proposto por Marco Dorigo em 1992. Em 2006, Kryzyof Socha desenvolveu o ACO para domínios contínuos (ACO). A ideia central do ACO é construir soluções baseadas na influência probabilística de cada componente (dimensão) do problema. Sua aplicação a problemas de otimização contínua tem sido considerada complexa, uma vez que o método de armazenamento de feromônio apresenta-se complexo para um número infinito de "pontos" (DORIGO; CARO, 1999). Versões recentes do algoritmo, tais como o ACO contínuo (CACO) (HUANG; HAO, 2006) abordaram este problema de forma diferente: o CACO é baseado em buscas locais e globais, sendo a busca global realizada por um algoritmo genético e a busca local implementada por um ACO. Métodos de otimização têm sido aplicados a diferentes problemas em sistemas de comunicação sem fio, entre eles o de detecção multiusuário DS-CDMA (MIRINELLO FILHO; SOUZA; ABRÃO, 2012), estimativa de parâmetros e alocação de recursos (potência, taxa e eficiência energética e espectral) (SAMPAIO et al., 2012), principalmente em sistemas cujos recursos computacionais e energéticos sejam limitados. Neste trabalho, a técnica de otimização heurística ACO é caracterizada inicialmente utilizando sete funções de benchmark clássicas, com diferentes graus de dificuldade de otimização, i.e. quantidade de ótimos locais e dimensão do problema (número de variáveis no domínio contínuo). Em seguida, os parâmetros de entrada do algoritmo ACO foram otimizados de forma não exaustiva, porém com resultados promissores; neste processo, geradores de números aleatórios com distribuição normal foram utilizados na amostragem das partículas (formigas) durante o processo de busca. Finalmente, o método ACO foi aplicado na solução do problema de controle de potência em sistemas CDMA.

O Algoritmo ACO

O ACO é uma meta-heurística baseada no comportamento de colônias de formigas em busca de comida. Na sua primeira versão (domínios discretos), o ACO foi aplicado a problemas de otimização combinatória (i.e., caixeiro viajante, roteamento, etc.). Cada formiga caminha entre os pontos do conjunto de entrada, e deposita feromônio a cada aresta que liga esses pontos (memória estigmergética). A seleção do próximo ponto é realizada de forma probabilística, levando em consideração a quantidade de feromônio na aresta, juntamente com a informação heurística (inverso do custo da aresta). Dado um conjunto de pontos vizinhos a uma formiga em um ponto *i* qualquer, o conjunto de probabilidades relativo à escolha de cada um desses pontos forma uma função distribuição discreta de probabilidade (PMF). A ideia fundamental do ACO é a substituição desta PMF por uma função contínua, ou seja, uma função

[1] Mecanismo de comunicação indireta entre seres, através de substâncias ou pistas deixadas no ambiente em que interagem. Pode ser vista também como a colaboração de agentes através de um meio físico, em sistemas descentralizados.

de densidade de probabilidade (PDF, Probability Density Function). Desta forma, uma formiga ao invés de selecionar um ponto qualquer situado em sua vizinhança, amostra um valor de uma PDF escolhida para a variável x_i (SOCHA; DORIGO, 2008). A Gaussiana é PDF mais utilizada neste processo. Por possuir apenas um ponto de máximo, uma única Gaussiana não é capaz de descrever uma situação onde duas regiões disjuntas de um espaço de busca são promissoras. Sendo assim o ACO utiliza um conjunto de Gaussianas unidimensionais para cada dimensão do problema. Cada conjunto de Gaussianas é definido por (SOCHA; DORIGO, 2008):

$$G^i(x) = \sum_{l=1}^{k} \omega_l g_l^i(x) = \sum_{l=1}^{k} \omega_l \frac{1}{\sigma_l^i \sqrt{2\pi}} e^{-\frac{(x-\mu_l^i)^2}{2\sigma_l^{i2}}} \qquad (1)$$

com $i = 1,...,n$, sendo n o número de dimensões do problema; $\boldsymbol{\omega}$ é o vetor de pesos, $\boldsymbol{\mu}^i$ é o vetor de médias, e $\boldsymbol{\sigma}^i$ o vetor de desvio padrão. Todos esses vetores possuem cardinalidade igual ao número de Gaussianas constituindo o conjunto em questão, ou seja, $|\boldsymbol{\omega}| = |\boldsymbol{\mu}^i| = |\boldsymbol{\sigma}^i| = k$. A quantidade de gaussianas por conjunto depende do tipo de problema a ser resolvido, e nem sempre deverá ser maior que o número de dimensões associado ao problema de otimização. No ACO é impossível armazenar as informações relativas ao feromônio em uma tabela, dado que existem infinitos pontos em um intervalo contínuo; portanto, infinitos caminhos a serem seguidos. Assim, utiliza-se um arquivo de soluções onde a cada iteração armazenam-se as soluções encontradas $\boldsymbol{s}_l$, e o valor das respectivas funções objetivo $f(\boldsymbol{s}_l)$. Tais soluções encontradas são utilizadas na geração dinâmica de PDF's, através de um método baseado no conjunto de soluções memorizadas. Os três vetores a partir dos quais cada conjunto de Gaussianas é parametrizado ($\boldsymbol{\omega}$, $\boldsymbol{\mu}^i$ e $\boldsymbol{\sigma}^i$) são calculados a partir das soluções no arquivo, para assim formar o conjunto de Gaussianas responsável por guiar as formigas no processo de busca, em cada dimensão do problema. O arquivo de soluções deve armazenar k soluções (quantidade de gaussianas em cada conjunto), e desta forma, o valor k determina a complexidade de cada conjunto. Para cada dimensão do problema é definido um conjunto diferente de gaussianas (G^i), e para cada uma delas, os valores da i- ésima variável de todas as soluções do arquivo tornam-se os elementos do vetor $\boldsymbol{\mu}^i$.

O peso ω_l da solução s_l é dado por:

$$\omega_l = \frac{1}{qk\sqrt{2\pi}} \exp\left[-\left(\frac{l-1}{qk\sqrt{2}}\right)\right] \qquad (2)$$

Ou seja, uma distribuição normal com média 1 e desvio padrão qk, onde q é um parâmetro do algoritmo. Para baixos valores de q, somente as melhores soluções são selecionadas gerando pouca diversidade, e para altos valores a probabilidade torna-se mais uniforme gerando mais diversidade. Desta forma, o parâmetro q no ACO é equivalente aos conceitos *best-so-far solution* e *iteration-best solution*, respectivamente. Quando o algoritmo é voltado para o *best-so-far* ele perde robustez para encontrar a solução ótima; por outro lado, torna-se mais rápido no processo de convergência. Contrariamente, quando a estratégia *iteration-best* é adotada, o ACO torna-se mais robusto, portanto resulta em convergência muito mais lenta. A robustez R é definida pela razão entre o complemento da porcentagem de falhas de convergência (F) do algoritmo, após N iterações, pelo número total de T realizações de teste de desempenho do algoritmo:

$$\mathsf{R} = 100\frac{1-F}{T} \quad [\%], \quad @\text{N Iterações} \qquad (3)$$

A velocidade do algoritmo é a quantidade média de iterações necessária para se atingir a convergência em termos de máximo erro quadrático médio (MSE) tolerável, considerando T realizações no teste.

Nesta seção foram apresentados os cálculos dos vetores $\boldsymbol{\mu}^i$ e $\boldsymbol{\omega}$, que parametrizam as misturas de PDF's. O cálculo do vetor $\boldsymbol{\sigma}^i$ é bem mais complexo e por isso será apresentado na próxima seção, de acordo com o processo prático do ACO.

O Framework Meta-heurístico ACO

Nesta seção discute-se o algoritmo ACO de acordo com a organização geral do princípio de otimização por colônia de formigas, recorrente em várias versões e implementação do algoritmo. O processo de implementação da técnica é apresentado passo-a-passo. Vale ressaltar que o cálculo do vetor $\boldsymbol{\sigma}^i$ não foi realizado na sessão anterior, devido à sua complexidade. A estrutura geral da técnica ACO (DORIGO; CARO, 1999), em suas várias versões é descrita por três etapas:

AntBasedSolutionConstruction(): Considerando as variáveis de decisão de cada solução X_i, $i = 1,...,n$, cada formiga constrói uma solução através de n passos. Sabendo-se que o ACO utiliza uma mistura de gaussianas em cada dimensão do problema, eq. (1), e que a quantidade de gaussianas em cada mistura é igual ao tamanho k do arquivo de soluções, conclui-se que em cada passo i a amostragem de será difG^inte. Assim, de acordo com (1), para que seja realizada uma amostragem é necessária a obtenção dos vetores $\boldsymbol{\mu}^i$, $\boldsymbol{\sigma}^i$, $\boldsymbol{\omega}$ da mistura. Dado que a obtenção do vetor $\boldsymbol{\mu}^i$ foi apresentada, discutiremos a obtenção do vetor $\boldsymbol{\sigma}^i$ de uma forma expedita para a obtenção de $\boldsymbol{\omega}_l$. Na prática, o processo de amostragem é realizado em três etapas: **a)** elementos do vetor $\boldsymbol{\omega}$ devem ser computados, observando-se que o argumento l em (2) (posição de cada solução) é constante ($1,..., k$), dado que ele sempre será o ranking das soluções. **b)** cada formiga deve escolher uma solução do arquivo, e a probabilidade de escolha deve ser dada de acordo com a normalização do peso de cada solução pela somatória de seus respectivos pesos:

$$p_l = \omega_l \left(\sum_{r=1}^{k} \omega_r \right)^{-1} \qquad (4)$$

ou seja, a probabilidade de cada solução ser escolhida pode ser resumida em um gerador de números aleatórios de distribuição normal, uma vez que a probabilidade de escolha de cada solução nunca mudará. Adotando esta estratégia, o primeiro passo do processo de amostragem não será mais necessário, bem como o vetor $\boldsymbol{\omega}_l$. **c)** amostragem passo-a-passo utilizando gerador de números aleatórios. A solução selecionada deve ser amostrada individualmente em cada dimensão (g_l^i, $i = 1...n$) , fazendo com que os parâmetros de cada mistura de gaussianas sejam visíveis apenas para uma única dimensão por vez, facilitando o cálculo do desvio padrão e possibilitando transformações lineares no problema sem alterar os resultados do algoritmo. Desta forma, apenas o valor $\boldsymbol{\sigma}_l^i$ precisa ser calculado a cada iteração, e não o vetor $\boldsymbol{\sigma}^i$ inteiro. Para estabelecer o valor do desvio padrão $\boldsymbol{\sigma}_l^i$ na iteração i, calcula-se a distância média entre a solução $\boldsymbol{s}_l$ escolhida, e todas as outras do arquivo, e multiplica-se pelo parâmetro ξ:

$$\sigma_l^i = \xi \sum_{e=1}^{k} \frac{| s_e^i - s_l^i |}{k-1} \qquad (5)$$

sendo $\xi > 0$ um parâmetro com valor igual para todas as dimensões e equivale à taxa de evaporação do feromônio, ou ao inverso da taxa de aprendizado, onde altos valores de ξ levam a uma convergência mais lenta, diferentemente do ACO convencional, onde o parâmetro ξ influencia na maneira em que a memória é utilizada. O algoritmo converge quando ξ chega à zero em todas as dimensões. ***PheromoneUpdate( ):*** o tamanho k do arquivo de soluções deve ser maior ou igual à quantidade de dimensões do problema. No início do algoritmo, o arquivo é inicializado com k soluções geradas a partir de uma distribuição uniforme. A partir disso, a atualização do feromônio é realizada adicionando as novas soluções, bem como removendo a mesma quantidade de piores soluções. Para que seja possível o tratamento de correlação entre as variáveis, capacitando o algoritmo a suportar transformações lineares no problema a ser otimizado, o tamanho k do arquivo deve ser maior que o número de dimensões do problema. O tamanho do arquivo é um dos principais responsáveis pela diversidade

do algoritmo (diversidade de soluções), e maior a possibilidade de escape de ótimos locais.

DaemonActions( ): é um componente opcional, utilizado para implementar ações centralizadas do algoritmo que não são acessíveis às formigas. Nesta parte do algoritmo, a solução encontrada deve ser atualizada e retornada como solução final. Além disso, é possível implementar métodos de busca local nesta seção, porém nesta versão, este recurso não é utilizado.

Resultados Numéricos

Nesta seção são apresentados os resultados de simulação objetivando avaliar a robustez e velocidade de convergência do ACO a partir da calibragem de parâmetros de entrada do algoritmo. O desempenho do algoritmo implementado é verificado em termos do número de iterações e percentagem de sucesso na convergência. A implementação do ACO deste trabalho está baseada no Algoritmo 1.

Testes de funções benchmark identificadas na Tabela 1 foram realizados considerando o número de avaliações da função custo durante o processo de otimização. Para todos os resultados apresentados, as soluções iniciais foram forçadas a assumirem condições desfavoráveis, tendo em vista verificar a capacidade de escape de ótimos locais do ACO.

A Tabela 2 sumariza a capacidade do algoritmo ACO em encontrar o ponto de ótimo global para diferentes funções e distintas dimensões. Para todos os problemas, os resultados apresentados são a média em $T = 103$ realizações. Nota-se que a diferença de desempenho em termos de percentagem de sucesso e/ou número de iterações (Itera) para o mesmo problema depende exclusivamente da natureza do problema, i.e., da quantidade de ótimos locais presentes na função e do intervalo de busca que contém a solução global. Por exemplo, a Fig. 1.a) sugere que a função Easom apresenta um único ponto de ótimo no intervalo, portanto, a percentagem de sucesso é de 100% para a função Easom de dimensão $n = 2$, 10 e 20; enquanto que o gráfico da Fig. 1.b) indica que a função Griewangk apresenta uma enorme quantidade de ótimos locais no intervalo, dificultando a convergência para o ótimo global. Duas outras funções benchmark e respectivas evoluções de convergência são apresentadas na Fig. 1.c) e 1.d). Ademais, valores para o erro quadrático médio MSE = $T^{-1} \Sigma_{t=1}^{T} |f_t(\mathrm{x}) - f(\mathrm{x}^*)|^2$ para quatro funções benchmark versus o numero de iterações são ilustradas na Fig. 2; T é o número de realizações adotado e $f(\mathrm{x}^*)$ o valor da função no ponto de ótimo. A Tabela II indica ainda que para a função Griewangk de dimensão $n = 2$, a percentagem de sucesso do ACO é de 100%. No entanto, quando $n = 10$, a esta percentagem é reduzida para 85% quando o intervalo de busca é [±5, 12]. Somente quando o intervalo de busca é reduzido para [±3], a percentagem de sucesso obtida torna-se total.

Algoritmo 1 ACO

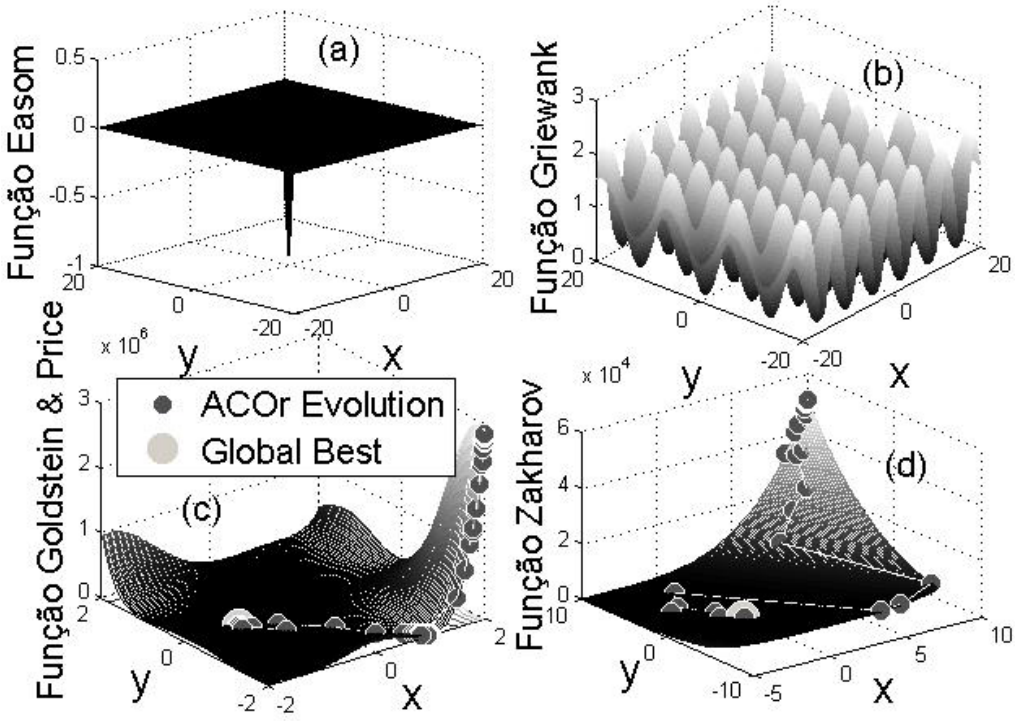
Require: It, $p_k[0] = \sigma_k^2$, $\forall k$
  for $i = 1$: It **do**
    for $k = 1 : K$ **do**
      Estime h_k, $\hat{I}_k$;
    end for
  end for
  Calcule γ_k e então compute K_{out}, onde $k \varepsilon K_{out}$ se $\gamma_k < \gamma_k^*$;
  if $\{K_{out}\} \neq \emptyset$ **then**
    tome o j-ésimo usuário com o pior ganho de canal em K_{out}
    defina $\gamma_j^* = 0$;
    retorne ao início.
  else return p_k^* $\forall_k$
  end if

Tabela 1 - Funções de Benchmark Utilizadas na Avaliação do ACO.

ID	Função Custo
Easom	$f_{es}(x) = -\cos(x_1)\cos(x_2)\exp(-(x_1-\pi)^2-(x_2-\pi)^2)$
Martin & Gaddy	$f_{MG}(x) = (x_1-x_2)^2+\left(\frac{x_1+x_2-10}{3}\right)^2$
Rosenbrok	$f_{Ro}(x) = \sum_{i=1}^{n-1} 100(x_i^2-x_i+1)^2+(x_i-1)^2$
Zakharov	$f_{Za}(x) = (\sum_{i=1}^{n} x_i^2)+\left(\sum_{i=1}^{n}\frac{ix_i}{2}\right)^2+\left(\sum_{i=1}^{n}\frac{ix_i}{2}\right)^4$
Griewangk	$f_{Gri}(x) = \left(\frac{1}{10}+\left(\sum_{i=1}^{n}\frac{x_i^2}{4000}-\prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right)+1\right)\right)^2$
Goldstein & Price	$f_{GP}(x) = [1+(x_1+x_2+1)^2(19-14x_1+13x_1^2-14x_2+6x_1x_2+3x_2^2)]\cdot[30+(2x_1-3x_2)^2(18-32x_1+12x_1^2-48x_2-36x_1x_2+27x_2^2)]$

Fonte: Kramer, Oliver: *Self-adaptative heuristics for evolutionary computation*. Springer. ISBN: 978-3-540-69280-5.

Figura 1 - Função a) Easom; b) Griewangk; c) Godstein & Price; d)Zakharov.



Fonte: Própria.

Tabela 2 - Resultados de Convergência para o ACO em $T=10^3$

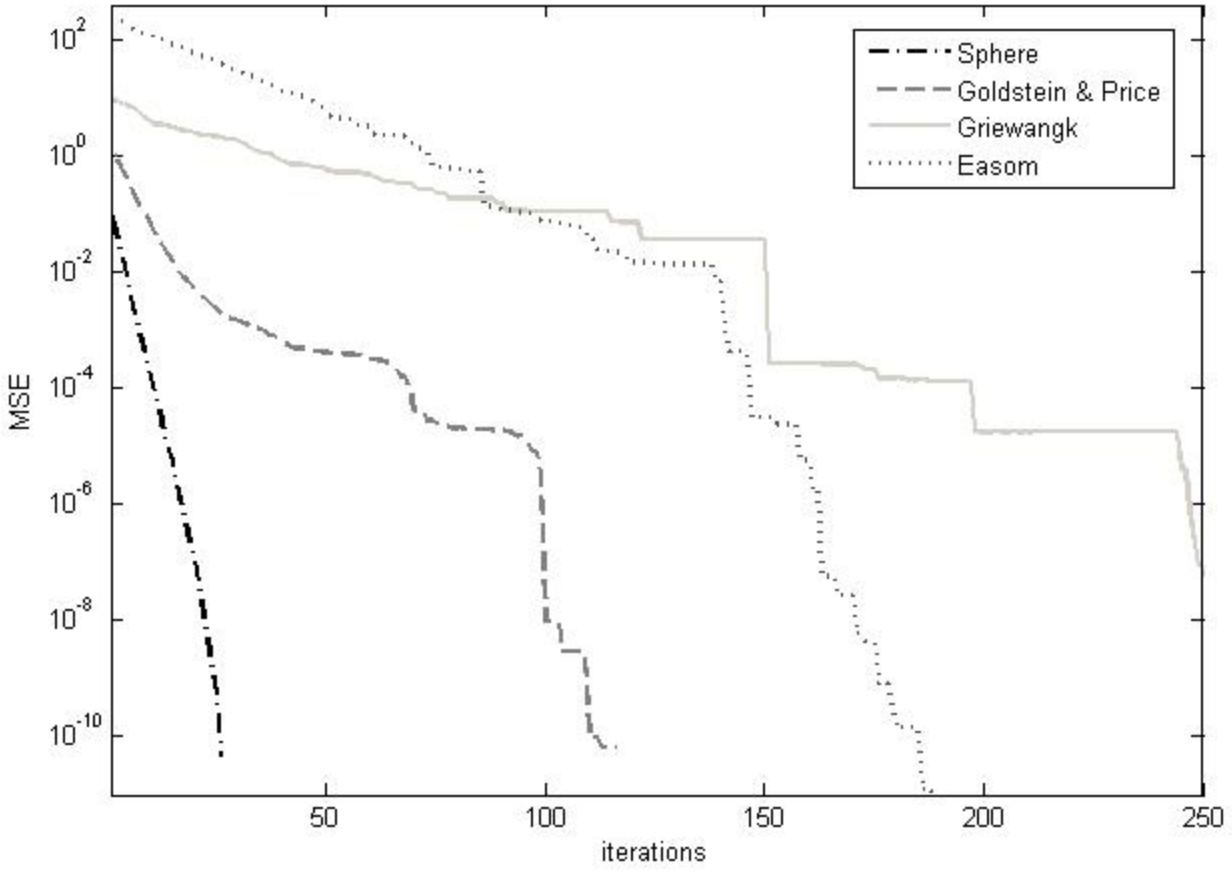
Função	*Itera*	*Dim*	*Intervalo*	*% Sucesso*
Easom	70	2	[-100;100]	100
Griewangk	189	2	[-5,12;5,12]	100
Goldstein Price	100	2	[-2;2]	100
Rosenbrok	90	2	[-5;10]	100
Sphere	26	2	[-5,12;5,12]	100
Zakharov	40	2	[-5;10]	100
Martin Gaddy	100	2	[-20;20]	100
Funções de Dimensão 10+				
Easom	341	10	[-100;100]	100
Easom	344	20	[-100;100]	100
Rosenbrok	517	10	[-5;10]	100
Rosenbrok	578	20	[-5;10]	100
Sphere	439	10	[-5,12;5,12]	100
Sphere	941	20	[-5,12;5,12]	100
Martin Gaddy	135	10	[-20;20]	100
Martin Gaddy	148	20	[-20;20]	100
Griewangk	2500	10	[-5,12;5,12]	85
Griewangk	281	10	[-3;3]	100

Fonte: Própria.

Robustez versus velocidade de Convergência

Funções multimodais constituem problemas complexos cuja obtenção do ótimo global pode tornar-se uma tarefa árdua para algoritmos de busca convencionais. Para algoritmos heurísticos, estas funções também constituem problemas complexos, principalmente se tais algoritmos não implementam estratégias elaboradas para escape de ótimos locais (mecanismos de diversidade). Desta forma, é necessário que os algoritmos tenham diversidade suficiente para não ficarem presos em ótimos locais, e ao mesmo tempo, que apresentem mecanismos de intensificação suficiente para evoluir caso encontrem a região próxima ao máximo global. Nitidamente, estes são objetivos contraditórios; de um lado, espera-se que o algoritmo heurístico venha a convergir o mais rápido possível, e de outro, que o mesmo não seja aprisionado em algum ótimo local. Mecanismos de diversificação-intensificação incluem: a) divisão do algoritmo em uma parte inicial de diversificação, e a final de intensificação; b) balanceamento dinâmico e adaptativo de seu comportamento, através da adoção de parâmetros que enfatizem a intensificação (convergência rápida) no início do processo, intensificando posteriormente os mecanismos de diversificação (convergência lenta) ao final do processo de convergência.

Figura 2 - Evolução do MSE para as funções-custo de duas dimensões (2D) Sphere, Goldstein & Price, Griewank e Easom; $T = 10^3$ realizações.



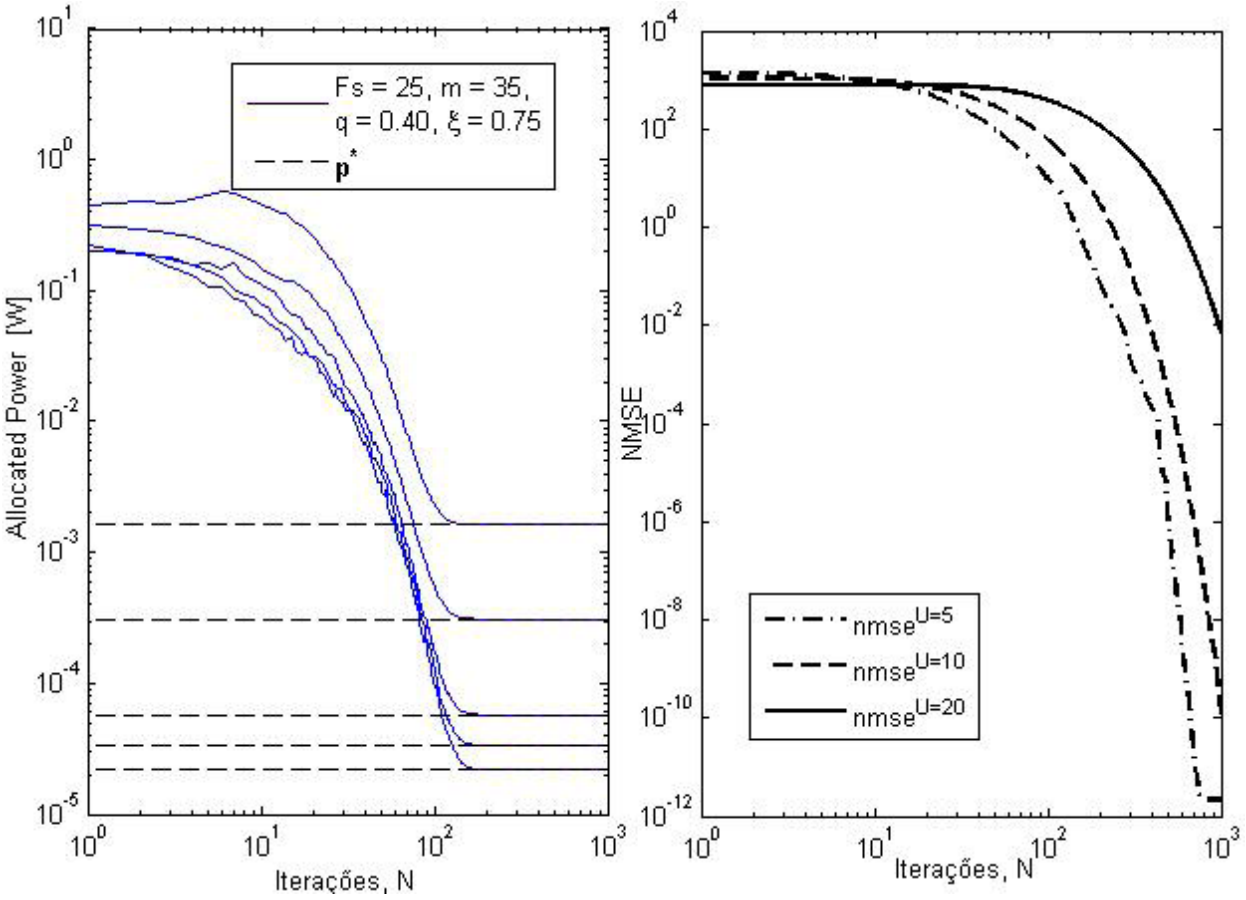
Fonte: Própria.

O ACO por sua vez, utiliza a técnica de balanceamento dinâmico. Sendo assim, quanto menor a taxa de aprendizado ξ^{-1}, na eq. (5) e maior o tamanho do arquivo (k), mais robusto será o algoritmo, com consequente aumento do tempo convergência. Isso mostra que o tamanho do arquivo *k*, influencia diretamente na diversidade e no tempo de convergência do algoritmo, pois quanto mais soluções houver no arquivo, mais iterações serão necessárias para que o desvio padrão σ^i, eq. (5), em cada dimensão para todas as soluções ($\sigma^i = [\sigma^i_1, \ldots, \sigma^i_l, \ldots, \sigma^i_k]$) tenda a zero. Além disso, o ACO possui o parâmetro *q*, também responsável pela diversidade do algoritmo. Considerando a eq. (2), quanto maior for o valor do parâmetro *q*, maior será o desvio padrão ($\sigma = qk$). Sabe-se que em uma variável aleatória Gaussiana, 68% das amostras situam-se no intervalo [±σ], ou seja, *qk*; enquanto 98% das amostragens aparecerão dentro do intervalo [±2σ] (SORENSEN, 2006). Desta forma, o valor do parâmetro *q* impacta na atualização de feromônio direcionada ou para o "melhor resultado desde o início do algoritmo" (intensificação, *best-so-far*), ou para o "melhor da iteração" (diversificação, *iteration-best*). Elevados valores de *q*, implicam em um espaço de busca constituído por um grande número de soluções possíveis. Assim, a busca é mais diversificada e o algoritmo torna-se mais robusto. Infelizmente, maior robustez significa maior tempo de convergência, portanto maior complexidade.

As figuras 3-a e 3-b apresentam valores de MSE e número de iterações em função dos parâmetros

q ε [0,0001;1] e ξ ε [0,7;1,5] para a função *Sphere* com 20 dimensões. O processo iterativo baseado em (FILHO; SOUZA; ABRÃO, 2012), consiste em obter a figura de mérito para cada valor de parâmetro de entrada do ACO enquanto os outros são fixados em uma configuração inicial. Para o problema *Sphere*, adotamos q = 0,1, ξ = 0,9, m = 20 e Fs = 8. Na figura 3-a é possível verificar que a robustez do algoritmo (MSE) melhora conforme o parâmetro q tende a seu valor máximo, enquanto é notável o aumento exponencial do número de iterações. Portanto, é possível utilizar diferentes abordagens no processo de calibragem deste parâmetro: pode-se encontrar um valor para o qual o algoritmo atinja um desempenho mínimo aceitável, reduzindo o tempo de convergência, ou adotar o valor máximo para q, e operar o ACO em sua forma mais poderosa, porém mais complexa.

Figura 3 - MSE e número de iterações em função dos valores atribuídos aos parâmetros q, considerando: a) primeiro passo de calibragem, bem como ξ b) no primeiro e c) segundo passo de calibragem, com q = 1.



Fonte: Própria.

Otimização dos Parâmetros de Entrada do ACO

A calibragem dos parâmetros de entrada de qualquer algoritmo heurístico é de fundamental importância no seu desempenho. Em geral, realiza-se uma varredura no intervalo definido para cada parâmetro, a fim de encontrar os melhores valores para o problema de otimização considerado. A otimização de parâmetros realizada neste trabalho é baseada em (FILHO; SOUZA; ABRÃO, 2012), na qual, inicialmente é considerada uma configuração de parâmetros obtida de forma não exaustiva. Em seguida, varia-se cada parâmetro ao longo de todo o intervalo definido, enquanto os outros parâmetros são fixados na configuração inicial. Com isso, obtêm-se os melhores valores para cada parâmetro, e assim, realiza-se este processo novamente em um intervalo menor, porém com maior precisão e assim sucessivamente.

Na figura 3-b, é visível que a melhor configuração para o parâmetro ξ encontra-se ao redor de $\xi = 1{,}3$, além disso, nota-se que o número de iterações aumenta consideravelmente em relação à ξ. Vale ressaltar que o algoritmo perde desempenho quando $\xi > 1{,}3$ devido ao número máximo de iterações adotado $N = 1000$, dado que o algoritmo apresentaria convergência muito mais lenta para esses valores. Finalmente, a figura 3-c mostra o segundo passo de calibragem do parâmetro ξ ao redor de $\xi = 1{,}3$, fixando $q = 1$. Evidencia-se o melhor valor deste parâmetro para a função Sphere-20D é $\xi = 1{,}26$.

Controle de Potência em redes CDMA

Nesta seção, o algoritmo ACO é aplicado na resolução do problema clássico de alocação de potência mínima necessária para garantir QoS a cada usuário de um sistema MPGDS/ CDMA (SAMPAIO et al., 2012). A taxa de erro de bit (BER) é usada como métrica de QoS, uma vez que esta está diretamente relacionada à relação sinal-ruído-mais-interferência (SNIR):

$$\gamma_i = \frac{F_i p_i \mid g_{ii} \mid^2}{\sum_{j=1, i\neq j}^{U} p_j \mid g_{ij} \mid^2 + \sigma^2}, i = 1, ..., U \tag{6}$$

sendo F_i é o ganho de processamento do i-ésimo usuário, p_i é a potência de transmissão, g_{ii} o ganho de canal (efeitos de desvanecimento de pequena e larga escala), g_{ij} o ganho de canal dos sinais interferentes e σ^2 a potência do ruído aditivo gaussiano branco (AWGN). O ganho de processamento F_i é dado pela razão entre a taxa de chip r_c e a taxa de informação mínima definida para o i-ésimo usuário r_i. A capacidade de Shannon para sistemas com espalhamento espectral em canal AWGN, considerando o gap entre o limite teórico e a taxa de informação real atingível é dada por:

$$r_i = \frac{w_i}{m_i} \log_2(1 + \theta_i \gamma_i), \left[\frac{bits}{\sec}\right] \tag{7}$$

sendo $w_i = W.F_i^{-1}$ a largura de banda do sinal não-espalhado do i-ésimo usuário, $m_i = \log_2 M_i$ a ordem de modulação utilizada e $W \approx r_c$ é a largura de banda do sistema. O problema de alocação de potência a ser resolvido via ACO (PA-ACO) é:

$$\begin{aligned} &\max J_1(\mathbf{p}) = \frac{1}{U}\sum_{i=1}^{U} \mathrm{F}_i^{th} \cdot \left(1 - \frac{p_i}{p_{\max}}\right), \\ &\text{s.t. (c.1)} \quad \gamma_i \geq \gamma_i^*; \quad \text{(c.2)} \quad 0 \leq p_i \leq p_{\max} \\ &\quad \text{(c.3)} \quad r_i = r_{i,\min}, \quad \forall i = 1, ..., U \end{aligned} \tag{8}$$

onde $\mathrm{F}_i^{\mathrm{th}} = \begin{cases} 1, & \gamma_i \geq \gamma_i^* \\ 0, & \text{c.c.} \end{cases}$ é a função de limiar (SAMPAIO et al., 2012). Os parâmetros utilizados nas simulações são apresentados na tabela 3. A Fig. 4-a ilustra a convergência típica para o problema PA-ACO em um sistema CDMA com $U = 5$ usuários. A Fig. 4-b apresenta os respectivos MSE normalizado (NMSE), obtidos para o mesmo problema considerando $U = \{5,$

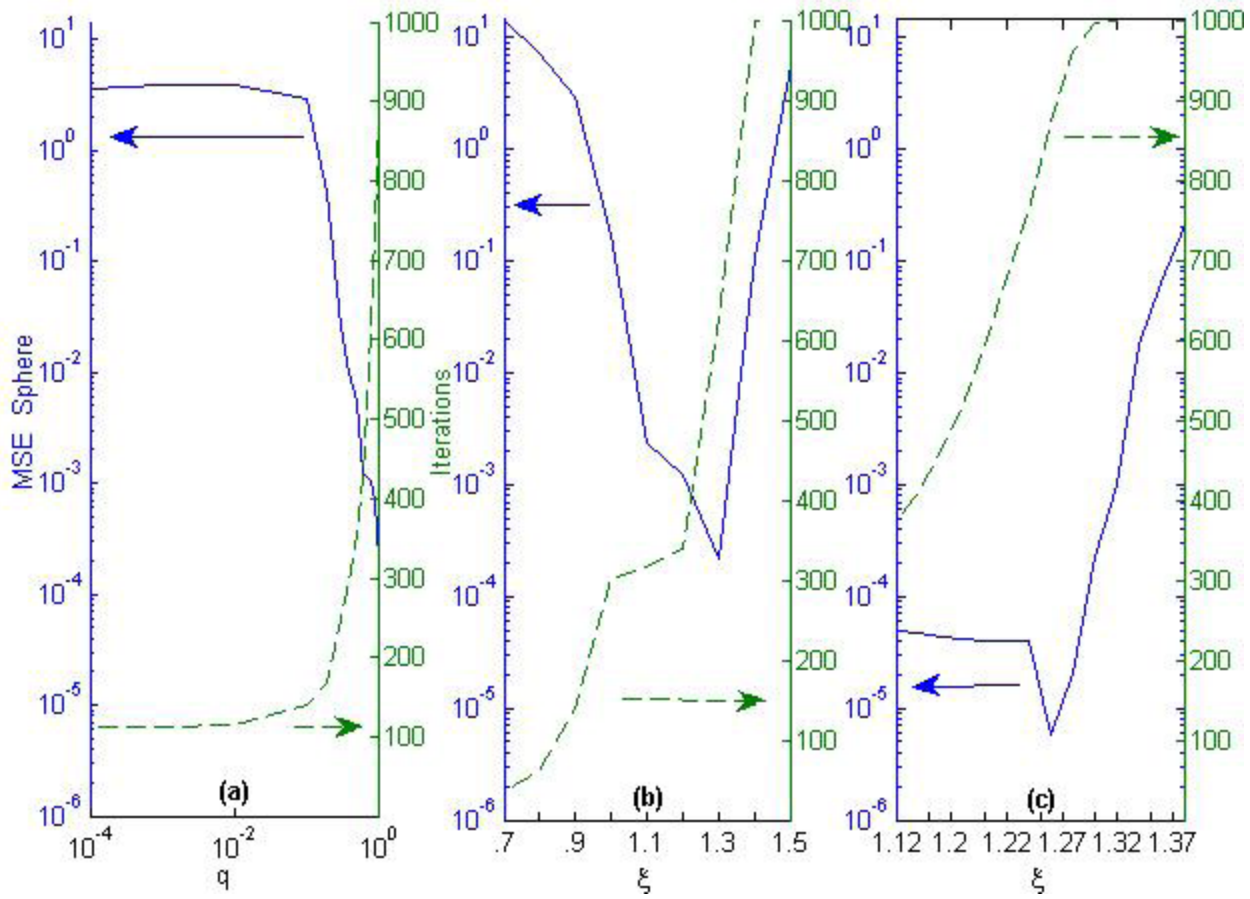
10, 20} usuários. É possível verificar que o PA-ACO é suficientemente robusto na resolução do problema de alocação de potência de sistemas CDMA.

Tabela 3 - Configurações do Sistema MPG-DS/CDMA

Parâmetro	Valor Adotado
Sistema MPG-DS/CDMA	
Potência de Ruído	P_n = -63 [dBm]
Taxa de *chip*	r_c=3,84×10^6 [cps]
Potência máxima por usuário	p_{max} = 1 [W]
# Estação Rádio Base	ERB = 1
Geometria da célula	Retangular, $x_{cell} = y_{cell}$ = 5 Km
Distribuição dos Terminais Móveis	~$U[x_{cell},y_{cell}]$
Usuários e QoS	
Serviços	[VOZ; VIDEO; DADOS]
Taxa dos usuários	$r_{i,min} = r_c \times$ [128;32;16]$^{-1}$ [bps]
BER alvo por usuário	[5×10^{-3}; 5×10^{-5}; 5×10^{-8}]
Algoritmo RA-ACO	
Tamanho do arquivo	Fs ε [8;25]
Fator de diversidade	q ε [0;1]
Taxa de evaporação do feromônio	ξ ε [0;1]
Tamanho da população	m ε [7;35]
Max. # de iterações	N = 1000

Fonte: Própria.

Figura 4 - Evolução do algoritmo RA-ACO para o problema de controle de potência. a) Evolução típica do vetor de potências em um sistema com U = 5 usuários. b) Evolução do NMSE para sistemas com U = {5, 10, 20} usuários.



Fonte: Própria.

Conclusões

Neste trabalho, o algoritmo ACO foi submetido a testes com várias funções benchmark, demonstrando robustez para problemas de dimensão 2D a 20D. Após realizar a otimização dos parâmetros de entrada do ACO, os resultados de otimização para o problema de controle de potência, em condições realísticas de configuração dos sistemas CDMA, mostraram total convergência, aliado à baixa complexidade do algoritmo ACO em relação aos métodos de inversão matricial. A menor complexidade implica em menor consumo de energia (em termos de processamento) enquanto mantém um desempenho próximo ao dos métodos determinísticos.

Referências

DORIGO, M.; CARO, G. Ant colony optimization: A new meta – heuristic. *Evolutionary Computation*, v. 2, p.1470-1477, CEC 99, IEEE, 1999.

MARINELLO FILHO, J. C.; SOUZA, R. N.; ABRÃO, T. Ant Colony Input Parameters Optimization for Multiuser Detection in DS/CDMA Systems. *Expert Systems with Applications*, v. 39, n. 17, p. 12 876 – 12 884, 2012. Disponivel em: http://www.sciencedirect.com/science/article/pii/S0957417412007038. Acesso em: 12/03/2014.

HUANG, H.; HAO, Z.; ACO for Continuous Optimization Based on Discrete Encoding. In: *Ant Colony Optimization and Swarm Intelligence. Lecture Notes in Computer Science, Springer Berlin Heidelberg*, v. 4150, p. 504-505, 2006. Disponível em: <http://dx.doi.org/10.1007/11839088_53>. Acesso em: 12/03/2014.

SAMPAIO, L. H. D.; MARQUES, M. P.; ADANIYA, M. H.; ABRÃO, T.; JESZENSKY, P. J. E (2013). Jeszensky (2013). *Ant Colony Optimization for Resource Allocation and Anomaly Detection in Communication Networks, Search Algorithms for Engineering Optimization*, Dr. Taufik Abrão (Ed.), ISBN: 978-953-51-0983-9, InTech, DOI: 10.5772/53338. Disponível em: http://www.intechopen.com/books/search-algorithms-for-engineering-optimization/ant-colony-optimization-for-resource-allocation-and-anomaly-detection-in-communication-networks. Acesso em: 12/03/2014.

SCHRIJVER, A. *Combinatorial Optimization, Polyedra and Efficiency*.Alemanha: Springer, 2003. Disponivel em: <http://www.springer.com/mathematics/applications/book/978-3-540-44389-6>. Acesso em: 12/03/2014.

SOCHA, K.; DORIGO, M. Ant colony optimization for continuous domains. *European Journal of Operation Research*, v.185, p. 1155-1173, 2008.

SORENSEN, L. *Introduction to the pratice of statistics*. Local: W. H. Freeman and Company, 2006. v. 1.

Recebido em 30 Julho 2013- Received on July 30, 2013.
Aceito em 10 Fevereiro, 2014 - Accepted on February 10, 2014.