Meta-Heurísticas e Otimização Combinatória: Aplicações em Problemas Ambientais

Article · January 2008

Source: OAI

CITATIONS
3

READS
2,037

9 authors, including:

José Carlos Becceneri
National Institute for Space Research, Brazil
57 PUBLICATIONS **372** CITATIONS
SEE PROFILE

Fernando Manuel Ramos
National Institute for Space Research, Brazil
221 PUBLICATIONS **2,811** CITATIONS
SEE PROFILE

Haroldo Fraga de Campos Velho
National Institute for Space Research, Brazil
475 PUBLICATIONS **3,255** CITATIONS
SEE PROFILE

Capítulo

2

Meta-heurísticas e Otimização Combinatória: Aplicações em Problemas Ambientais

José Carlos Becceneri

Resumo

Este minicurso oferece uma introdução aos problemas de natureza combinatória e à aplicabilidade de algumas meta-heurísticas utilizadas atualmente na solução desses problemas. Inicialmente, analisamos alguns conceitos básicos mais comumente empregados na área de otimização, utilizando o Problema do Caixeiro Viajante. Também são abordadas as meta-heurísticas Simulated Annealing e Colônia de Formigas. Concluímos este minicurso expondo algumas aplicações a problemas ambientais nas quais foram utilizadas meta-heurísticas. Nosso objetivo com esse estudo é oferecer uma abordagem introdutória para pessoas que queiram se iniciar na área de Otimização Combinatória e Meta-heurísticas.

1. Introdução

Problemas de otimização consistem em achar a melhor combinação dentre um conjunto de variáveis para maximizar ou minimizar uma função, geralmente chamada de função objetivo ou função custo. Esses problemas podem ser divididos em três categorias: aqueles cujas variáveis assumem valores reais (ou contínuos), aqueles cujas variáveis assumem valores discretos (ou inteiros) e aqueles em que há variáveis inteiras e contínuas, classificados, respectivamente, como problemas de Otimização Contínua, Otimização Combinatória ou Discreta, e Otimização Mista. Nesta seção estudaremos apenas problemas de Otimização Combinatória.

Nosso objetivo com esse estudo é oferecer uma abordagem introdutória para pessoas que queiram se iniciar na área de Otimização Combinatória e Meta-heurísticas. Para exemplificar alguns conceitos básicos, utilizaremos o Problema do Caixeiro Viajante, definido na próxima seção. Duas meta-heurísticas foram escolhidas para o nosso estudo: *Simulated Annealing* e Colônia de Formigas. Na penúltima seção, daremos alguns exemplos da aplicabilidade das meta-heurísticas aqui estudadas. Escolhemos problemas aplicados às questões ambientais, por serem um assunto de relevante importância para a

sociedade não só brasileira, como mundial. Porém, é importante ressaltar que podemos aplicar essas meta-heurísticas em diversos outros tipos de problemas tais como de astronomia, medicina, engenharia, etc.

2. Conceitos

Nesta seção definiremos diversos conceitos comumente empregados na área de Otimização Combinatória. Para exemplificar alguns desses conceitos, utilizaremos o Problema do Caixeiro Viajante, por ser um problema clássico e de fácil entendimento.

2.1 O Problema do Caixeiro Viajante (PCV)

Esse problema é estabelecido do seguinte modo:

Um caixeiro viajante deve visitar *n* cidades e retornar à cidade de onde partiu, sendo que, com exceção da cidade inicial, cada cidade deve ser visitada apenas uma vez. O objetivo é minimizar o custo dessa viagem.

Uma representação muito usada para este problema consiste em colocá-lo na forma de um grafo, onde as cidades são os nós e os caminhos entre elas são os arcos. Consideremos o exemplo a seguir com 4 cidades, com as coordenadas cartesianas mostradas na Tabela 1. Calculando-se as distâncias cartesianas entre essas cidades, obtemos os valores mostrados na Tabela 2.

cidades	x	y
A	1	1
B	3	2
C	2	4
D	5	5

Tabela 1: coordenadas cartesianas de 4 cidades.

Distâncias	A	B	C	D
A	0	2.236	3.162	5.657
B	2.236	0	2.236	3.606
C	3.162	2.236	0	3.162
D	5.657	3.606	3.162	0

Tabela 2: distâncias entre as cidades da Tabela 1.

A Figura 1 representa essas cidades na forma de um grafo.

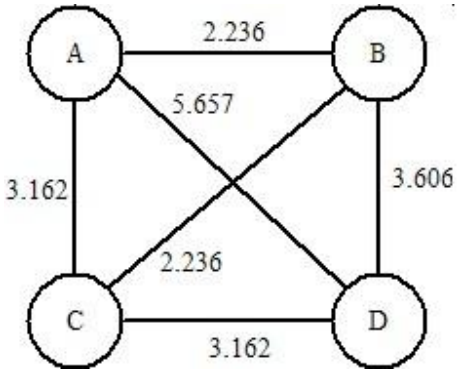

Figura 1: PCV da Tabela 1 na forma de grafo

Para detalhes sobre o surgimentos desse problema, que data de 1832, e sobre as muitas aplicações que ele pode ter, consulte [LAWLER et al, 1985].

2.2 Problemas de Otimização

Seja f uma função com domínio S. Chamaremos essa função de função custo, função de avaliação ou função objetivo. Dizemos que temos um problema de minimização quando, dado f, queremos encontrar $s^* \in S$ tal que $f(s^*) \leq f(s)$, $\forall\ s \in S$. Se o objetivo é encontrar um $s^* \in S$ tal que $f(s^*) \geq f(s)$, $\forall\ s \in S$, então temos um problema de maximização.

Frequentemente, ao invés dos termos minimização ou maximização, usamos o termo **otimização**, sempre que não houver dúvidas sobre o objetivo do problema em estudo (isto é, se queremos encontrar um valor que minimize ou maximize).

Por exemplo, em relação ao PCV, dizemos que queremos otimizar a viagem, significando que: a) queremos minimizar a distância percorrida (nesse caso, a função f mede distância); ou b) queremos maximizar o lucro da viagem (nesse caso, f avalia o lucro obtido com a viagem).

2.3 Espaços de Buscas

Um espaço de busca é um conjunto que contém todas as soluções do problema que estamos tentando resolver. Esse espaço pode ser finito ou infinito enumerável. Sempre consideraremos que esse espaço contenha apenas soluções viáveis, isto é, aquelas que obedecem às restrições do problema. Exemplifiquemos esses conceitos.

Consideremos um PCV com 5 cidades rotuladas de A, B, C, D e E. Uma solução – isto é, uma rota – do tipo C, E, A, B, D, C é viável, pois cada cidade é visitada apenas uma única vez e, então, retorna-se à cidade inicial. Uma rota do tipo A, B, C, D, B, E, A não representa uma solução viável, pois a cidade B é visitada 2 vezes.

Designaremos por $N(s)$ um conjunto chamado de "vizinhos do ponto s". Esse conjunto é constituído por pontos de S que podem ser considerados *próximos* de s. Se quisermos formalizar essa definição, podemos estabelecer uma métrica d e um número real ε, definindo $N(s) = \{s' \in S / d(s, s') \leq \varepsilon\}$.

2.4 Algoritmos

Um algoritmo é um processo sistemático para a resolução de um problema. Podemos classificar os algoritmos de diversos modos, alguns deles, definidos a seguir. Uma observação cabe neste ponto: não é nosso objetivo analisar uma taxonomia de algoritmos e métodos computacionais; apenas apresentaremos algumas das classificações mais usadas atualmente na área de Otimização.

Algoritmos podem ser classificados como exatos ou aproximados. Algoritmos exatos, tais como técnicas de enumeração do tipo *branch-and-bound* – veja, por exemplo, [ZIVIANI, 2004] ou [ARENALES et al, 2006] – garantem encontrar uma solução ótima para qualquer instância de um problema de otimização. Muitos desses problemas são *NP*-Difícil [GAREY e JOHNSON, 1979], implicando que métodos exatos poderiam levar um tempo de computação muito grande para determinados tamanhos de entrada. Por isso, nas últimas três décadas, muita atenção tem sido dada a métodos aproximados.

Também podemos classificar os algoritmos em determinísticos e não-determinísticos.

Algoritmos determinísticos são aqueles nos quais, sempre que a entrada do problema for repetida, o resultado produzido será o mesmo. Já os algoritmos não-determinísticos, também chamados de probabilísticos, podem, diante de uma mesma entrada do problema, produzir resultados diferentes, pois consideram no seu processamento algum evento pseudo-aleatório, como, por exemplo, a geração de um número pseudo-aleatório. Para um estudo sobre a geração de números pseudo-aleatórios em computador, consulte [KNUT, 1998].

2.5 Heurísticas e Meta-heurísticas

O nome **Heurística** é derivado da palavra grega *heuriskein,* que significa descobrir. Hoje esse termo é usado para descrever um método "que, baseado na experiência ou julgamento, parece conduzir a uma boa solução de um problema, mas que não garante produzir uma solução ótima" [FOULDS, 1984]. Também podemos considerar esse termo como associado a um conhecimento circunstancial, não verificável, nem matematicamente verificável.

Já o termo "meta-heurística" deriva da composição de duas palavras gregas: "heurística", já explicada no parágrafo anterior, e o prefixo "meta", que significa "após", indicando um nível superior de descoberta.

É comum encontrarmos a expressão "heurística gulosa", ou "*Greedy Heuristic*". Essa heurística resolve um problema de otimização procurando, a cada iteração, o elemento constituinte da solução que mais reduz o custo total naquele momento. A busca se encerra quando todos os elementos da solução tiverem sido calculados. A heurística gulosa apresenta a vantagem de ser rápida e, em alguns casos, produzir soluções eficientes. Por exemplo, considerando os dados expressos pela Figura 1, se aplicarmos uma heurística gulosa, deve-se escolher um nó inicial. Esse nó inicial para a ser o nó corrente e, a partir dele, escolher, como próximo nó, aquele que possui menor distância do nó corrente. Esse procedimento é aplicado até que todas as cidades tenham sido visitadas.

Muitas definições podem ser dadas sobre o termo "meta-heurística" na computação, como, por exemplo: "Uma meta-heurística é um conjunto de conceitos que podem ser

usados para definir métodos heurísticos, os quais podem ser aplicados a um conjunto amplo de diferentes problemas. Em outras palavras, uma meta-heurística pode ser vista como uma ferramenta algorítmica geral que pode ser aplicada a diferentes problemas de otimização, com modificações relativamente pequenas para torná-la adaptável a um problema específico" [Metaheuristics Network].

Uma meta-heurística é uma estratégia de busca, não específica para um determinado problema, que tenta explorar eficientemente o espaço das soluções viáveis desse problema. São algoritmos aproximados que incorporam mecanismos para evitar confinamento em mínimos ou máximos locais. Conhecimentos específicos do problema podem ser utilizados na forma de heurística para auxiliar no processo de busca (por exemplo, na busca de um possível bom vizinho de um determinado ponto). Resumindo, podemos dizer que meta-heurísticas são mecanismos de alto nível para explorar espaços de busca, cada uma usando um determinado tipo de estratégia.

De grande importância na aplicabilidade de uma meta-heurística é o balanço dinâmico entre *diversificação* e *intensificação*, fazendo uma distinção entre os termos ingleses *exploration* e *exploitation*. O primeiro pode-se traduzir por **diversificação**, **exploração diversificada**, **busca em largura** ou simplesmente **exploração**; o segundo por **exploração focada, busca em profundidade ou intensificação**. Um dos desafios na aplicação de uma meta-heurística é encontrar o equilíbrio ideal entre diversificação e intensificação.

A estratégia de busca de uma meta-heurística depende da filosofia empregada por ela, e seu objetivo é escapar dos mínimos locais a fim de proceder a exploração do espaço de busca por soluções ainda melhores. Na Figura 2, os pontos *a* e *b* podem ser considerados mínimos locais e o ponto *c*, mínimo global entre os pontos *x* e *y*.

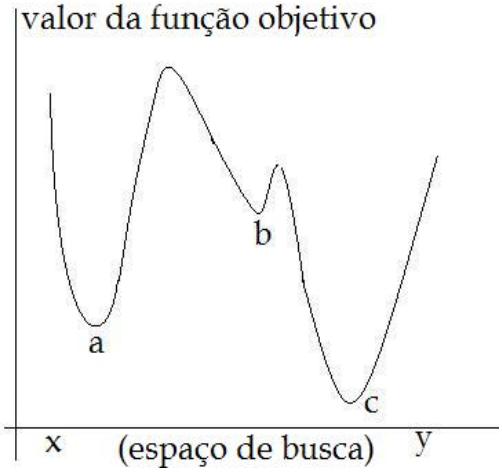

Figura 2: Mínimos Locais e Mínimo Global.

Classificação das Meta-heurísticas

Há diferentes modos de se classificar e descrever uma meta-heurística. A seguir são apresentados alguns aspectos que podem ser levados em consideração.

2.5.1 Inspiradas ou não na natureza

Talvez o modo mais intuitivo de se classificarem meta-heurísticas seja com base nas origens do algoritmo. Nesse sentido, podemos diferenciar os algoritmos inspirados na natureza, como os Algoritmos Genéticos [HOLLAND, 1975] e os Algoritmos das Formigas [Bonabeu et al, 1999], daqueles não-inspirados na natureza, como a meta-heurística *Iterated Local Search* [RAYWARD-SMITH et al, 1996]. Essa classificação pode não ser muito significativa, visando mais a oferecer uma abordagem didática. Por exemplo, o uso de memória na Busca Tabu [GLOVER, 1989] é ou não inspirada em um processo natural?

Dizemos que uma meta-heurística é bio-inspirada quando suas regras de busca tentam simular alguns aspectos do comportamento de seres vivos como, por exemplo, o Algoritmo das Formigas, o algoritmo dos Pássaros [EBERHART E SHI, 2001] ou o algoritmo das Abelhas [ABBASS, 2001].

2.5.2 Baseadas em população ou baseadas em busca de pontos simples

No método construtivo ou de trajetória, parte-se de um conjunto solução vazio e segue-se acrescentando elementos a esse conjunto até obter uma solução viável para o problema. Métodos populacionais partem de um conjunto de soluções iniciais (a população inicial) e tentam encontrar uma solução melhor alterando-se elementos dessa população.

Um exemplo de modelo construtivo é o Algoritmo das Formigas, pois descreve uma trajetória no espaço de busca durante a sua execução.

Meta-heurísticas baseadas em população, ao contrário, executam um processo de busca modificando a população inicial, como os Algoritmos Genéticos e o *Simulated Annealing*.

3. Casos de estudo

Nesta seção, apresentaremos duas das meta-heurísticas comumente empregadas nos dias atuais, que exemplificam os conceitos acima estudados.

O *Simulated Annealing* (SA) é uma meta-heurística inspirada na natureza e é também um método populacional. Traduções usuais para esse nome são: Têmpera Simulada, Recozimento Simulado ou Arrefecimento Simulado.

A Otimização por Colônia de Formigas (*Ant Colony Optimization*) é uma meta-heurística bio-inspirada e é também um método de trajetória.

3.1 Simulated Annealing (SA)

SA é uma das mais antigas meta-heurísticas que incorporaram um mecanismo para escapar de mínimos locais. Teve sua origem no algoritmo de Metropolis usado em mecânica estatística e foi apresentado como um algoritmo de busca para problemas de otimização. Para um estudo mais detalhado desse processo físico, consulte [KIRKPATRICK, 1983], o artigo seminal desta técnica. A idéia fundamental é permitir movimentos que resultem em soluções de pior qualidade do que a da solução corrente, a fim de escapar de mínimos locais. Isto é feito simulando-se o processo físico de recozimento (*annealing*). A probabilidade de fazer um movimento desse tipo decresce durante a busca. A Figura 3 esquematiza um algoritmo desse tipo. A nomenclatura utilizada é a seguinte:

s é a solução gerada na iteração corrente

*s** é a melhor solução encontrada

f é a função custo

T_0 é a temperatura inicial

T é a temperatura corrente

a é um número real, entre 0 e 1, gerado aleatoriamente.

O pseudocódigo pode ser expresso por:

Atribui à s uma solução inicial 1

s*= s 2

T = T_0 3

Enquanto condições de fim não são entradas 4

s´= EscolhaVizinho(N(s)) 5

Δ = f(s´) - f(s) 6

Se Δ < 0 então

s = s´ 7

if(f(s') < f(s*)) s*= s' 8

senão

Gera(a) 9

if (a < exp(-Δ/T)) s= s' 10

Fim_se

atualiza T 11

atualiza outras condições de fim 12

Figura 3: Pseudocódigo do Algoritmo *Simulated Annealing*

É interessante discorrermos sobre cada linha numerada do algoritmo acima (as linhas não numeradas não necessitam de explicação).

Na linha 1, obtém-se uma solução inicial, o que pode ser feito de forma aleatória através de algum outro método.

Na linha 2, atribui-se a *s** o valor da solução inicial *s*, por ser a melhor solução conhecida até este passo.

Na linha 3, atribui-se a *T* o valor da temperatura inicial (T_0). O parâmetro T_0 deve ser suficientemente grande para que todas as transições sejam inicialmente aceitas. Evidentemente, esse valor depende do tipo do problema e da instância analisada. Na literatura, existem muitas propostas para o cálculo da temperatura inicial (veja, por exemplo, [AARTS e KORST, 1989]). Citamos aqui duas sugestões: $T_0 = ln\, f(s^0)$, onde $f(s^0)$

é o custo da solução inicial; ou $T_0 = \Delta E_{max}$, onde ΔE_{max} é a diferença máxima de custo entre duas soluções vizinhas. No último caso, porém, o cálculo pode consumir muito tempo computacional; por isso, freqüentemente, usamos uma estimativa para T_0.

Na linha 4, é estabelecido um critério de parada. Das muitas possibilidades, três delas são:

até que se atinja um número máximo de iterações, definido no início do processamento;

até que a temperatura T atinja um determinado valor;

até que se atinja um determinado número de melhoras da função objetivo (isto é, cada vez que na linha 8 se testar um valor verdadeiro, incrementa-se um contador de melhoras).

Na linha 5, escolhe-se um vizinho da solução corrente *s*. Essa função de escolha é essencial ao bom desempenho do algoritmo, pois, se analisarmos muitos vizinhos, podemos comprometer o tempo de processamento (veja a discussão sobre intensificação e diversificação feita na seção 2.5). Para o número de elementos a serem testados, as seguintes alternativas podem ser utilizadas: um número constante de vezes; uma função da temperatura, por exemplo, 1/log(T); ou até que um determinado número de aceitações ou rejeições tenha sido atingido.

Na linha 6, calcula-se a diferença entre os valores da solução corrente e do novo ponto encontrado na vizinhança (*s'*).

Na linha 7, se o valor da função no ponto *s'* for menor do que em *s*, então *s'* passa a ser a solução corrente.

Na linha 8, verifica-se se o valor corrente é menor que o valor armazenado em *s**. Em caso afirmativo, *s** recebe o valor de *s'*.

Na linha 9, caso o valor de *f(s')* seja maior que o valor de f(s), gera-se um número aleatório entre 0 e 1, indicando que uma solução pior foi encontrada em *s'*.

Na linha 10, aceita-se essa solução pior como solução corrente, com a probabilidade indicada pela fórmula. Essa é uma tentativa de se escapar de mínimos locais.

Na linha 11, determina-se que a temperatura seja atualizada a cada iteração. Diversas estratégias podem ser empregadas, conforme [GOLDSTEIN e WATERMAN, 1988]. Citamos duas:

variação geométrica: $T(t) = T(t) \, . \, \alpha(t)$, onde $\alpha(t)$ pode ser constante;

variação logarítmica: $T(t) = c/(\ln(1+t))$, onde *c* é uma constante.

Na linha 11, outras condições de fim também devem ser atualizadas, se existirem, como, por exemplo, incrementar o número de iterações.

Consideremos o PCV exemplificado na Figura 1. Mostremos como os passos descritos anteriormente podem ser executados. As seguintes decisões devem ser definidas a priori.

O critério de parada. Vamos estabelecer que ele será dado por um número máximo de iterações, armazenado na variável *loopmax*. A variável *nloop* indica o número da iteração corrente.

A escolha de um vizinho: uma permutação será gerada a partir da solução corrente.

A cada iteração, a temperatura será decrescida de 10%.

Considerando o algoritmo descrito na Figura 3 e o PCV descrito na Figura 1, poderíamos ter a seguinte seqüência de ações, descritas na Figura 4:

nloop= 0

s ← ABDCA // essa solução tem custo 12,168

s*= s // armazena a melhor solução

$T = T_0 = \ln(12,168) = 2,50$ // um critério para T_0

Enquanto *nloop* < *loopmax*

s´= EscolhaVizinho(N(s))= ABCDA

$\Delta = f(s') - f(s) = f(ABCDA) - f(ABDCA) = 13,291 - 12,168 = 1,123$

$\Delta > 0$

Gera(*a*) = 0,4

$0,4 < \exp(-\Delta/T) = \exp(-1,123/2,5) = 0,638$

s= ABCDA

T = 0,9 * T // decréscimo de 10% na temperatura

nloop= nloop+1

Figura 4: execução de uma iteração do algoritmo SA.

A partir deste ponto, o algoritmo deve ser executado o número de vezes indicado em *loopmax*. Depois disso, a solução armazenada em s^* representará a melhor solução encontrada.

Um exemplo de utilização do AS em conjunto com o Algoritmo Genético pode ser encontrado em [RANCK et al, 2007].

3.2 Colônia de Formigas

Colônia de Formigas é uma meta-heurística baseada no comportamento de um grupo de formigas tentando achar comida. Neste texto, essa meta-heurística será representada pela sigla ACO (*Ant Colony Optimizatiom)*.

Muitos aspectos do comportamento de um grupo de insetos são estudados pelas teorias de *Self-Organization* (SO), que tentam explicar como indivíduos de comportamento simples podem construir uma sociedade complexa. Por exemplo, sabemos que em tais sociedades há grupos de trabalhadores especializados e, aparentemente, não há um gerente operacional. Então, como explicar a complexidade das tarefas realizadas? Como a cooperação acontece? Como escrito em [CAMAZINE et al, 2003], “estruturas complexas resultam da iteração de comportamentos surpreendentemente simples executados por indivíduos possuindo apenas informação local”.

Para entendermos como o ACO foi criado, é necessário conhecermos um pouco sobre o comportamento das formigas. Cada formiga, quando caminha em busca de alimento, deposita no solo uma substância química chamada de feromônio. Na possibilidade de seguir por vários caminhos, uma formiga pode, para decidir entre eles, considerar a quantidade de feromônio – depositado anteriormente por outras formigas – em cada um. Um caminho com muito feromônio é mais atrativo que outro com menos. Com o decorrer do tempo, essa substância química sofre evaporação. O que se nota na natureza é que, após um determinado tempo, a maioria das formigas decidiu trilhar um bom caminho (talvez o mais otimizado) entre o seu ninho e a fonte de comida (caminho em destaque na Figura 5).

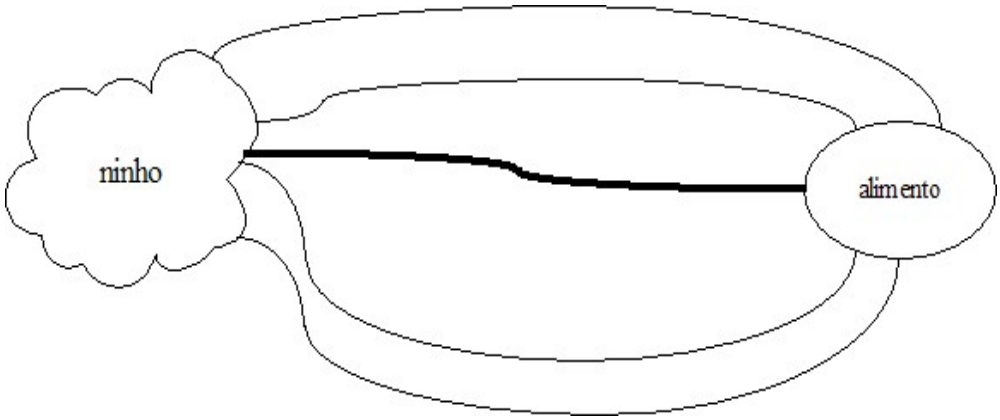

Figura 5: caminhos trilhados por formigas entre seu ninho e uma fonte de comida.

O algoritmo de otimização por formigas foi proposto por Marco Dorigo, em 1992, em sua tese de doutorado [Dorigo, 1992]. A idéia básica de todo o algoritmo baseado em formigas é o uso de um mecanismo conhecido como reforço positivo (*positive feedback*) [Dorigo et al, 1991], baseado na analogia com o comportamento de certas espécies de formigas que, como já mencionado, derramam nos caminhos por elas trilhados uma substância química chamada de feromônio, possibilitando o reforço dos caminhos mais trilhados, os quais são, possivelmente, os melhores. Um feromônio virtual é utilizado para manter as boas soluções na memória do computador. Também há o conceito de reforço negativo, implementado através da analogia com o processo de evaporação que o feromônio sofre na natureza. A combinação do reforço positivo (depósito de feromônio) com o negativo (evaporação), permite que se evite, na maioria dos casos, uma convergência prematura do algoritmo para soluções, possivelmente não ruins, mas talvez, longe da ótima. O comportamento cooperativo é outro conceito importante aqui: algoritmos de colônia de formigas fazem uso da exploração simultânea de diferentes soluções por meio de formigas idênticas. As melhores formigas influenciam a exploração das demais, através das estratégias empregadas para atualizar o feromônio nos caminhos.

Para compreendermos o funcionamento dessa meta-heurística, vamos considerar sua aplicação ao PCV, conforme exemplificado na Figura 1.

Cada formiga pode ser considerada um agente que:

ao se mover do nó *i* para o nó *j*, pode depositar feromônio no caminho sendo percorrido, dependendo da estratégia de depósito sendo utilizada. Em geral, há duas possibilidades: ou

todas as formigas depositam feromônio nos caminhos por elas percorridos, ou apenas a melhor formiga de cada iteração deposita feromônio no caminho percorrido;

possui uma tabela *J* com as cidades que pode visitar. Isso é necessário para evitar que uma formiga vá para uma cidade por onde ela já passou;

para selecionar um nó a ser visitado, usa uma função de probabilidade. Uma formiga *k*, estando no nó *i*, calcula sua probabilidade de ir para o nó *j* na iteração *t* através da expressão:

$$p_{ij}^{k}(t) = \frac{\left[\tau_{ij}(t)\right]^{\alpha} \left[\eta_{ij}\right]^{\beta}}{\sum_{l \in J_i^k} \left[\tau_{il}(t)\right]^{\alpha} \left[\eta_{il}\right]^{\beta}} \quad ,$$

onde:

τ_{ij} é a quantidade de feromônio entre os nós *i* e *j* no instante *t*;

η_{ij} é o inverso da distância entre os nós *i* e *j*. Esse parâmetro é chamado de *desejabilidade*. No caso do PCV, quanto maior a distância entre os nós, menor é o desejo de ir para esse nó; quanto menor essa distância, maior o desejo.

A cada iteração:

o feromônio depositado em cada arco sofre uma evaporação (isto é, seu valor é decrementado por uma constante de evaporação). Seu valor é decrementado por uma constante de evaporação f_e, $0 \leq f_e < 1$. Uma possível equação que pode ser utilizada para realizar essa atualização é:

$$f_{ij}(t) \leftarrow (1 - f_e).\, f_{ij}(t) + f_e.f_0$$

Outras estratégias tanto para realizar o depósito, quanto para realizar a evaporação podem ser definidos. Vide [BLUM, 2005] para uma discussão ampla sobre esse assunto.

dependendo do método de atualização utilizado, ou o melhor caminho encontrado ou todos os caminhos percorridos sofrem um acréscimo de feromônio.

A complexidade do ACO, utilizando a notação *big*-O, é dada por $O(t.n^2.m)$, onde:

t é o número de iterações;

n é o número de nós;

m é o número de formigas.

Para um estudo sobre como expressar a complexidade de um algoritmo, consulte [SZWARCFITER e MARKENZON, 1994].

Para aplicar adequadamente o ACO a um problema devemos:

ter uma representação apropriada do problema na forma de grafo;

saber definir a *desejabilidade*;

saber como obedecer às restrições do problema;

ter uma regra de atualização do feromônio;

definir uma regra de transição probabilística.

A seguir os passos para a implementação do ACO para um problema de otimização são apresentados.

Passo 1. Inicialização do sistema:

a) Calcular a desejabilidade de cada nó;
b) Definir a quantidade de feromônio de cada arco;
c) Distribuir as formigas de modo aleatório entre os nós.

Passo 2. Para cada iteração definida:

Cada formiga percorre todos os nós do grafo, calculando, em cada nó, sua probabilidade de movimento.

Passo 3. Verificar qual formiga obteve a melhor rota.

Passo 4. Atualização do feromônio:

a) Incrementar a quantidade de feromônio na melhor rota;
b) Decrementar quantidade de feromônio nas demais rotas.

Consideremos o PCV exemplificado na Figura 1. Mostremos como os passos acima podem ser executados.

Consideremos uma solução gulosa iniciando no nó A, dada por ABDCA, cujo custo é 13,2913. Definamos as variáveis *Lnn*= 13,2913 e *nc* = 4 (número de cidades).

Temos que definir uma matriz de feromônio de dimensão 4*x*4, que será representada por *F*. Cada elemento dessa matriz é inicializado com o valor τ_0, obtido, segundo sugestão encontrada em [BONABEAU et al, 1999] através da expressão:

$\tau_0 = 1.0/(nc*Lnn) = 1/(4*13{,}2913) = 1/53{,}1652 = 0{,}019$.

Então, a matriz de feromônio, *F*, é dada por:

$$F = \begin{bmatrix} 0{,}000 & 0{,}019 & 0{,}019 & 0{,}019 \\ 0{,}019 & 0{,}000 & 0{,}019 & 0{,}019 \\ 0{,}019 & 0{,}019 & 0{,}000 & 0{,}019 \\ 0{,}019 & 0{,}019 & 0{,}019 & 0{,}000 \end{bmatrix}$$

Os *loops* não são permitidos, por isso, os elementos da diagonal principal são zerados (a probabilidade de se estar em um nó e nele permanecer durante uma iteração é zero).

Mais uma matriz é utilizada, como mostra a Tabela 2 (matriz de distâncias).

Dois vetores devem ser definidos: um para armazenar a melhor rota encontrada na iteração corrente (VC) e um para armazenar a melhor rota encontrada dentre todas as iterações realizadas (VT). Esses vetores serão inicializados com algum valor adequado para representar que eles ainda não foram utilizados.

Consideremos *nf* como o número de formigas. Esse valor é escolhido heuristicamente. Geralmente, se não há nenhuma indicação contrária, fazemos *nf* = *nc*. Para simplificar os cálculos que serão mostrados, utilizaremos *nf*= 2. Identificaremos essas 2 formigas por 0 e 1.

Essas *nf* formigas são espalhadas aleatoriamente pelos nós. Suponhamos que 2 formigas sejam colocas no nó A e nenhuma nos demais nós.

Como critério de parada, podemos estabelecer um tempo máximo de processamento ou um número fixo de iterações. Neste exemplo, será realizada 1 iteração completa das 2 formigas.

A formiga 0, consultando a sua tabela de cidades que ainda não visitou (J={B, C, D}), irá calcular as seguintes probabilidades na iteração 0:

$p_{AB}^{0}(0)= 0{,}48$

$p_{AC}^{0}(0)= 0{,}34$

$p_{AD}^{0}(0)= 0{,}18$

(repare que a soma das 3 probabilidades acima é 1)

Dadas essas 3 probabilidades, a formiga 0 deve decidir para qual cidade ela irá. Para isso, é utilizada a seguinte estratégia: sorteia-se um número entre 0 e 1. Se este número for menor ou igual a 0,48, a cidade B será a escolhida; se for maior que 0,48 e menor ou igual a 0,82, será a cidade *C*; caso contrário, com 18% de chances, irá para a cidade D.

Suponhamos que o número sorteado seja 0,5. Então, a formiga 0 vai para a cidade C. A tabela de cidades que ainda não foram visitadas é atualizada: J={B, D}.

Estando na cidade C, ela deve decidir entre ir para B ou D. As seguintes probabilidades são calculadas:

$p_{CB}^{0}(0)= 0{,}41$

$p_{CD}^{0}(0)= 0{,}59$

Novamente, através de um sorteio, a formiga 0 decidirá se irá para B ou D. Suponhamos que a cidade sorteada, com 59% de chances, seja a D.

Então temos a seguinte rota até agora: ACD. A partir desse ponto não precisamos mais calcular probabilidades, pois a única alternativa possível é escolher a cidade B e, em seguida, retornar a cidade A. Então o percurso completo feito pela formiga 0 é: ACDBA, cujo custo é: 12,166.

Agora é a vez da formiga 1 traçar a sua rota. A formiga 1 irá calcular as seguintes probabilidades (sempre verificando a tabela de cidades que ainda não foram visitadas):

$p_{AB}^{1}(0)= 0,48$

$p_{AC}^{1}(0)= 0,34$

$p_{AD}^{1}(0)= 0,18$

Dada essas 3 probabilidades, a formiga 1 deve decidir para qual cidade ela irá. A estratégia para tal decisão é a mesma utilizada pela formiga 0.

Suponhamos que o número sorteado seja 0,1. A formiga 1 vai para a cidade B, que é retirada da lista de cidades que ela pode visitar.

Estando na cidade B, ela deve decidir entre ir para C ou D. As seguintes probabilidades são calculadas:

$P_{BC}^{1}(0)= 0,38$

$P_{BD}^{1}(0)= 0,62$

Novamente, através de um sorteio, a formiga 0 decidirá entre ir para C ou D. Suponhamos que a cidade sorteada, com 38% de chances, seja a C.

Então temos a seguinte rota até agora: ABC. A partir desse ponto, não precisamos mais calcular probabilidades, pois a única alternativa possível é escolher a cidade D e, em seguida, retornar à cidade A. Então o percurso completo feito pela formiga 0 é dado por ABCDA, cujo custo é: 13,291.

A melhor rota obtida foi a percorrida pela formiga 0. Essa rota será armazenada em uma estrutura de dados que indica a melhor solução encontrada na iteração corrente. O caminho percorrido pela formiga 0 será reforçado com uma quantidade adicional de feromônio. Também é importante ressaltar que seguimos a estratégia de fazer o reforço de feromônio apenas no caminho percorrido pela formiga vencedora. Porém, existem outras estratégias, como permitir que todas as formigas depositem uma quantidade de feromônio nas suas rotas. Para um estudo detalhado sobre essas possibilidades, consulte [BLUM, 2005].

Neste ponto, dois parâmetros devem ser utilizados, para:

incrementar o caminho percorrido pela formiga vencedora – definamos um aumento de 0,1 (10%) nesse caminho;

decrementar os caminhos não trilhados – definamos uma diminuição de 0,10 (10%) nesses caminhos (poderia ser um valor diferente do definido para o incremento).

A quantidade de feromônio a ser depositada ou retirada em cada iteração é um dos pontos a serem estudados por quem está implementando o algoritmo. Pode ser uma quantidade fixa, como no exemplo acima dado, ou dependente de algum fator, como, por exemplo, a qualidade da solução.

Então, a matriz de feromônio *F* passa a ser:

$$F = \begin{bmatrix} 0{,}000 & 0{,}0209 & 0{,}0209 & 0{,}0171 \\ 0{,}0209 & 0{,}000 & 0{,}0171 & 0{,}0209 \\ 0{,}0209 & 0{,}0171 & 0{,}000 & 0{,}0209 \\ 0{,}0171 & 0{,}0209 & 0{,}0209 & 0{,}000 \end{bmatrix}$$

Agora devemos armazenar a melhor solução encontrada na iteração corrente em VC (vetor que armazena a melhor rota da iteração corrente).

Para finalizar o processamento de uma iteração, devemos comparar as soluções armazenadas em VC e VT. VT deve conter, sempre, a rota com o melhor custo. Como estamos finalizando a primeira iteração, fazemos VT= VC.

Finalizado este ponto, devemos iniciar a próxima iteração. Agora as formigas 0 e 1 irão realizar novamente os passos indicados acima, porém com uma matriz *F* diferente da situação inicial.

Quando todas as iterações tiverem sido feitas, teremos armazenado em VT a melhor rota encontrada durante o processamento.

Existem muitas variantes para esse algoritmo, dentre as quais escolhemos uma para explicar sua funcionalidade básica. Em [BONABEAU et al, 1999] e [BLUM, 2005], podem ser encontradas diversas outras variantes.

Para aplicar o ACO no PCV, fizemos:

a representação do problema na forma de grafo (nós são cidades e arcos são caminhos);

a definição da *desejabilidade* como o inverso da distância;

uma tabela com as cidades já visitadas;

uma regra heurística de atualização do feromônio, tanto para a evaporação como para o incremento nos caminhos percorridos;

a definição de uma função de probabilidade.

Essa meta-heurística tem sido muito empregada atualmente. Suas idéias conceituais são simples e os resultados encontrados em muitos problemas práticos são muito bons. Veja, por exemplo, [BECCENERI e ZINOBER, 2001], [CARVALHO et al, 2007] e [BECCENERI e SANDRI, 2006].

Nas referências [Becceneri e Sandri, 2006] e [Becceneri et al 2008] é apresentada uma contribuição teórica dada pelos autores do trabalho, que tem mostrado excelentes resultados quando utilizada: é o depósito de feromônio em caminhos adjacentes do melhor caminho trilhado por uma formiga.

4. Aplicações a Problemas Ambientais

Nesta seção, citaremos alguns trabalhos que utilizam meta-heurísticas em aplicações ambientais. Como já ressaltada na Introdução deste trabalho, podemos ter aplicabilidade em diversos outros tipos de problemas, tais como de astronomia, de medicina e de engenharia, pois, por sua própria definição, a meta-heurística possui um caráter genérico, isto é, não é específica a algum tipo de problema. Escolhemos mostrar a sua aplicabilidade a problemas ambientais por ser este um tema de relevante importância no momento atual da nossa sociedade, ao qual o Instituto Nacional de Pesquisas Espaciais (INPE) possui diversos pesquisadores dedicados.

Quatro trabalhos foram escolhidos para exemplificar a aplicação das meta-heurísticas aqui estudadas.

1) "An Inverse Formulation for Diffusive Bridgman Growth Using Ant Colony Optimization in a High Performance Environment", apresentado no XXV CNMAC (Congresso Nacional de Matemática Aplicada e Computacional), em Nova Friburgo, Rio de Janeiro, no período de 16 a 19 de setembro de 2002. Autores: Nanci N. Arai, Roberto P. Souto, Airam J. Preto, Haroldo F. Campos Velho, José Carlos Becceneri, Maurício Fabbri, Stephan Stephany.

Neste estudo, foi utilizado ACO em um problema de crescimento de cristais. Chamamos de cristais todo tipo de material no qual os átomos se distribuem de forma organizada e regular. Como explicado em [BOSCHETTI], "Os materiais cristalinos, além de propriedades interessantes, podem ser processados em laboratório com elevado grau de pureza. Sua estrutura cristalina pode ser reproduzida através de técnicas especiais conhecidas como técnicas de crescimento de cristais". O ACO foi aplicado para se avaliar o valor de uma determinada função (coeficiente de difusão). O modelo foi numericamente implementado por um método de diferenças finitas. Apesar de ser um problema de otimização contínua, obtivemos sucesso com a aplicação do algoritmo das formigas.

2) On the Use of the Ant Colony System for Radiative Properties Estimation, R. P. SOUTO, S. STEPHANY, J. C. BECCENERI, H. F. CAMPOS VELHO, and A. J. SILVA NETO, em Proceedings of the 5th International Conference on Inverse Problems in Engineering: Theory and Practice, Cambridge, UK, 11-15th July 2005.

Problemas inversos de transferência radioativa de calor possuem várias aplicações em muitas diferentes áreas, tais como astronomia, ciências ambientais, engenharia e medicina [Chedin et al, 2003], [Craig e Brown, 1986], [Hakim e McCormick, 2003], [Miesch 2003], [Siewert, 2002]. Típicos exemplos são estimações de funções e parâmetros para modelos climáticos globais, ótica hidrológica e tomografia computadorizada [Arridge, 1999] [Velho et al, 2003a], [Velho et al, 2003b], [Gao, 1990], [Hochberg et al, 2003], [Zhou et al, 2002].

Quando formulados implicitamente, problemas inversos são usualmente escritos como problemas de otimização para os quais utilizamos algoritmos baseados em enxame inteligentes [Bonabau et al, 1999], tais como o ACO.

Na década de 1990, o ACO foi aplicado com sucesso a inúmeros problemas de otimização combinatória [Dorigo et al, 1996], e mais recentemente tem sido utilizado para

estimar parâmetros reais associados a problemas inversos [Becceneri e Zinober, 2001], [Preto et al, 2004], [Souto et al, 2004].

Neste trabalho, o ACO foi aplicado em um problema inverso de transferência radioativa para se determinar a espessura ótica, o albedo de espalhamento simples e as refletividades difusas nas superfícies internas das extremidades de um meio (para um entendimento desses termos, veja [Bell e Glasstone, 1970]). Mais detalhadamente, aplicamos o ACO para minimizar a função objetivo que determinava as propriedades físicas aqui citadas.

3) ESTIMATING ATMOSPHERIC AREA SOURCE STRENGTH THROUGH PARTICLE SWARM OPTIMIZATION, Eduardo F. P. da Luz, Haroldo F. de Campos Velho, José C. Becceneri, Débora R. Roberti, IPDO2007 - INVERSE PROBLEMS, DESIGN AND OPTIMIZATION SYMPOSIUM, April 16-18, Newport Beachside Resort Hotel, 16701 Collins Avenue, Sunny Isles Beach (Miami Beach), Florida 33160, U.S.A.

O trabalho acima foi conseqüência de uma dissertação de mestrado defendida no INPE em 2007 [LUZ, 2007] cujo resumo é apresentado a seguir.

A poluição atmosférica é um problema de aspecto global que leva cientistas e líderes mundiais a juntar esforços com o objetivo de obter soluções de médio e longo prazo para evitar problemas futuros. Uma das ações necessárias envolve a localização de fontes e a estimação da quantidade de emissão de poluentes. Atualmente, técnicas de problemas inversos podem ser utilizadas neste processo, e quando este problema é visto sob a ótica de um problema de otimização com restrições, o uso de técnicas alternativas de pesquisa operacional, tais como algoritmos bio-inspirados, podem ser consideradas como uma alternativa viável e de comprovada robustez. Esta dissertação apresenta os resultados obtidos pela utilização de um algoritmo bio-inspirado (PSO) e comparados com um método exato para a localização e estimação de fontes de poluição atmosférica.

O trabalho acima utilizou a meta-heurística PSO, inspirada na coreografia que um bando de pássaros executa em vôo. Novamente, como nos dois trabalhos citados acima, uma função objetivo foi definida e, então, utilizou-se o PSO para achar um ponto ótimo do domínio dessa função.

4) INVERSE PROBLEMS, DESIGN AND OPTIMIZATION (IPDO-2007) Symposium in Miami Beach, Florida, FUZZY ANT COLONY OPTIMIZATION FOR ESTIMATING CHLOROPHYLL CONCENTRATION PROFILE IN OFFSHORE SEA WATER. Adenilson C. Carvalho, Haroldo F. de Campos Velho, Stephan Stephany, Roberto P. Souto, José C. Becceneri, Sandra Sandri.

A estimação de algumas propriedades óticas pode ser estudada através da concentração de clorofila nos oceanos. Para este trabalho, um problema inverso foi formulado. Uma equação de transferência radioativa foi estabelecida. Uma função objetivo foi formulada, através das diferenças quadráticas entre dados experimentais e computacionais. Inicialmente, utilizou-se o ACO padrão (como explicado neste capítulo). A seguir, utilizou-se o ACO versão *fuzzy* [BECCENERI e SANDRI, 2006]. Os resultados com essa nova versão foram bem melhores, implicando que a diferença entre os valores obtidos a partir de dados experimentais e computacionais foi muito pequena.

5. Conclusões

Fizemos, neste pequeno estudo, uma introdução às meta-heurísticas aplicadas a problemas de otimização combinatória, onde demos alguns exemplos de sua aplicabilidade a problemas ambientais. Cabe ressaltar que meta-heurísticas também têm sido amplamente utilizadas para problemas de otimização contínua, onde as variáveis são reais. Por exemplo, em [BECCENERI e SANDRI, 2006], utiliza-se o ACO, com algumas modificações, para encontrar mínimos de funções. Uma meta-heurística adequada para problemas de otimização contínua é a inspirada no comportamento de um bando de pássaros durante o vôo, chamada de *Particle Swarm Optimization* (PSO). Para exemplos da utilização dessa técnica, consulte [LUZ et al, 2007] e [BECCENERI et al, 2006], onde o PSO é aplicado a problemas ambientais (especificamente, ao estudo de dispersão de poluentes).

Bibliografia

AARTS, E.; KORST, J. **Simulated Annealing and Boltzmann Machines.** John Wiley & Sons Editores, 1989.

ABBASS, H. A. MBO: **Marriage in Honey Bees Optimization, A Haplometrosis Polygynous Swarming Approach.** In Proceedings of the IEEE Congress on Evolutionary Computation, pages 207--214, 2001.

ARENALES, M.; ARMENTANO, V.; MORABITO, R.; YANASSE, H. **Pesquisa Operacional para cursos de Engenharia.** Elsevier Editora Ltda. Ed. 2006.

ARRIDGE, S. R. **Optical tomography in medical imaging**. *Inverse Probl*ems (1999) **15**, R41-R93.

BECCENERI, J. C; SANDRI, S. **Function optimization using ant colony systems with pheromone dispersion**. Proceedings of thr XI International Conference on Information, Processing and Management of Uncertainty in Knowledge-based Systems (IPMU), França, 2006.

BECCENERI, J. C; SANDRI, S., LUZ, E.F.P. **Using Ant Colony Systems with Pheromone Dispersion in the Travelling Salesman Problem**, Proceedings of the 11th International Conference of the Catalan Association for Artificial Intelligence, Espanha, 2008.

BECCENERI, J. C.; STEPHANY, S.; VELHO, H. F. C; SILVA, A. J. **Solution of the Inverse Problem of Radiative Properties Estimation with the Particle Swarm Optimization Technique**. INVERSE PROBLEMS, DESIGN AND OPTIMIZATION (IPDO-2006) Symposium, Florida, EUA, 2007. Proceedings in CD-Rom.

BECCENERI, J. C.; ZINOBER, A. **Extraction of Energy in a Nuclear Reactor**. XXXIII Simpósio Brasileiro de Pesquisa Operacional, Campos do Jordão, SP, 2001. Anais do XXXIII Simpósio Brasileiro de Pesquisa Operacional, em CD-ROM.

BELL, G.; GLASSTONE, S. ***Nuclear Reactor Theory***. New York: Van Nostrand Reinold, 1970

BLUM, C. **Ant colony optimization: Introduction and recent trends**. Physics of Life Reviews, v. 2, p. 353–373, 2005.

BONABEAU E.; DORIGO, M.; THERAULAZ, G. **From natural to artificial swarm intelligence**. Oxford University Press, 1999.

BOSCHETTI, http://www.las.inpe.br/~cesar/Infrared/materiais.htm, visitado em 27/12/2007.

CAMAZINE, S.; DENEUBOURG, J. L.; FRANKS, N. R.; SNEYD, J.; THERAULA, G.; BONABEAU, E. **SELF-ORGANIZATION IN BIOLOGICAL SYSTEMS**. Princeton University Press, 2003.

CARVALHO, A. C.; VELHO, H. F. C.; STEPHANY, S.; SOUTO, R. P.; BECCENERI, J. C.; SANDRI, S. **FUZZY ANT COLONY OPTIMIZATION FOR ESTIMATING CHLOROPHYLL CONCENTRATION PROFILE IN OFFSHORE SEA WATER**. Inverse Problems, Design and Optimization Symposium, Florida, EUA, 2007.

CHEDIN, A.; SERRAR, S.; HOLLINGSWORTH, A.; ARMANTE, R.; SCOTT, N. A. Detecting annual and seasonal variations of CO_2, CO and N_2O from a multi-year collocated satellite-radiosonde data-set using the new rapid radiance reconstruction (3R-N) model. Journal of Quantitative Spectroscopy and Radiative Transfer (2003) 77, 285-299.

CRAIG, I. J. D.; BROWN, J. C. Inverse Problems in Astronomy: A Guide to Inversion Strategies for Remotely Sensed Data. Adam Hilger Ltd., Bristol, 1986.

DORIGO, M. **Ottimizzazione, Apprendimento Automático, Ed Algoritmi Basati su Metafora Naturale**, Ph.D. Dissertation, Politécnico di Milano, Italy, 1992.

DORIGO, M.; MANIEZZO, V.; COLORNI, A. **Positive Feedback as a Search Strategy**, Tech. Report, No. 91-016, Politécnico di Milano, Italy, 1991.

DORIGO, M.; MANIEZZO, V.; COLORNI, A. **The ant system: optimization by a colony of cooperating agents**. *IEEE T. on Syst. Man Cy. B* (1996) **26** (2), 29–41.

EBERHART, R. C. AND SHI, Y. **Particle swarm optimization: developments, applications and resources.** Proc. congress on evolutionary computation 2001 IEEE service center, Piscataway, NJ., Seoul, Korea., 2001.

FOULDS, L. R. **Combinatorial Optimization for Undergraduates**. Springer-Verlag, New York, 1984, p. 114.

GAO, F.; NIU, H.; ZHAO, H.; ZHANG, H. **The forward and inverse models in time-resolved optical tomography imaging and their finite-element method solutions**. *Image and Vision Computing* (1998) **16**, 703-712.

GAREY, M. R.; JOHNSON, D. S. Computers and Intractability: a Guide to the Theory of NP-Completeness, W. H. Freeman, 1979.

GLOVER, F. **Tabu Search, part I**. ORSA Journal on Computing, v. 1, n. 3, p. 190-206, 1989.

GOLDSTEIN, L.; WATERMAN, M. **Neighborhood Size in the Simulated Annealing Algorithm**. American Journal of Mathematical and Management Sciences, v. 8, nos. 3-4, 409-423, 1988.

HAKIM, A. H.; MCCORMICK, N. J. **Ocean optics estimation for absorption, backscattering, and phase function parameters.** *Appl. Optics* (2003) **42**, 931– 938.

HOCHBERG, E.J.; ATKINSON, M. J.; ANDRÉFOUËT, S. **Spectral reflectance of coral reef bottom-types worldwide and implications for coral reef remote sensing**. *Remote Sensitive. Environ.* (2003) **85**, 159-173.

HOLLAND, J. H. **Adaptation in Natural and Artificial Systems**. University of Michigan Press, 1975.

KIRKPATRICK , S.; GELLAT, D. C.; VECCHI, M. P. **Optimization by Simulated Annealing**. Science, v. 220, p. 671-680, 1983.

KNUT, D. E. **The Art of Computer Programming**. Addison-Wesly Professional, vol. 1-3 Boxed Set, 2ª ed., 1998.

LAWLER, E. L.; LENSTRA, J. K.; KAN, A. H. G. R.; SHMOYS, D. B. **The Traveling Salesman Problem.** Chichester. A Wiley_interscience Publication, John Wiley & Sons, 1985.

LUZ, E. F. P. Estimação de fonte de poluição atmosférica usando otimização por enxame de partículas. **Dissertação (Mestrado em Computação Aplicada) – Instituto Nacional de Pesquisas Espaciais, São José dos Campos, 2007.**

LUZ, E. F. P.; VELHO, H. F. C.; BECCENERI, J. C.; ROBERTI, D. R. **ESTIMATING ATMOSPHERIC AREA SOURCE STRENGTH THROUGH PARTICLE SWARM OPTIMIZATION**. INVERSE PROBLEMS, DESIGN AND OPTIMIZATION (IPDO-2007) Symposium, Florida, EUA, 2007. Proceedings in CD-Rom.

Metaheuristics Network, http://www.metaheuristics.net/index.php?main=1, visitado em 26/11/2007.

MIESCH, C.; CABOT, F.; BRIOTTET, X.; HENRY, P. Assimilation method to derive spectral ground reflectance of desert sites from satellite datasets. *Remote Sens. Environ.* (2003) 87, 359-370.

PRETO, A. J.; VELHO, H. F. C.; BECCENERI, J. C.; FABBRI, M.; ARAI, N. N.; SOUTO, R. P.; STEPHANY, S. **A new regularization technique for an ant-colony based inverse solver applied to a crystal growth problem,** *13th Inverse Problems in Engineering Seminar (IPES-2004)*, Cincinnati, USA, 14-15 June, 2004, pp. 147-153.

RANCK, R.; BECCENERI, J. C.; SILVA, J. D. S. Extração de energia em um reator nuclear utilizando *Simulated Annealing* e Algoritmos Genéticos. In: SIMPÓSIO DE PESQUISA OPERACIONAL E LOGÍSTICA DA MARINHA, 10, 2007, Rio de Janeiro, RJ.

RAYWARD-SMITH, V. J.; OSMAN, I. H.; REEVES, C. R.; SMITH, G. D. **Modern Heuristic Search Methods**. John Wiley & Sons Ltd, Baffins Lane, Chichester, West Sussex, England, 1996.

SIEWERT, E. **Inverse solutions to radiative-transfer problems based on the binomial or the Henyey-Greenstein scattering law**. Journal of Quantitative Spectroscopy and Radiative Transfer. *Transfer* (2002) **72**, 827-835.

SOUTO, R. P.; VELHO, H. F. C.; STEPHANY, S.; SANDRI, S. **Reconstruction of chlorophyll concentration profile in offshore ocean water using ant colony system**. *First Hybrid Metaheuristics (HM-2004)*, Valencia, Spain, 22-23 August, 2004, pp. 19-24.

SZWARCFITER**,** J. L.; MARKENZON, L. **Estruturas de Dados e Seus Algoritmos.** Editora LTC, 1994, cap. 1.

VELHO, H. F. C.; VILHENA, M. T.; RETAMOSO, M. R.; PAZOS, R. P. **An application of the LTS_N method on an inverse problem in hydrologic optic**s. Progress in Nuclear Energy (2003a) **42**, 457-468.

VELHO, H. F. C.; RAMOS, F. M.; CHALHOUB, E. S.; STEPHANY S.; CARVALHO, J. C.; SOUZA, F. L**. Inverse problems in space science and technology**, *Proceedings of the 5th International Conference on Industrial and Applied Mathematics - ICIAM*, Sidney, Australia, 7-11 July, 2003b.

ZIVIANI, N. **Projeto de Algoritmos com implementações em Pascal e C**. Editora Thomson Pioneira, 2004, cap. 9.

ZHOU, H. C.; HOU, Y. B.; CHEN, D. L.; ZHENG, C. G. **An inverse radiative transfer problem of simultaneously estimating profiles of temperature and radiative parameters from boundary intensity and temperature measurements**. Journal of Quantitative Spectroscopy and Radiative Transfer (2002) **74**, 605-620.