Capítulo 4

Otimização por Colônia de Formigas

Mauro Henrique Mulati*, Ademir Aparecido Constantino
e Anderson Faustino da Silva

Resumo: Este capítulo apresenta uma fundamentação teórica e aplicações da meta-heurística Otimização por Colônia de Formigas. São apresentados algoritmos de aplicações desta meta-heurística para resolução do Problema do Caixeiro Viajante e do Problema de Cobertura de Conjunto. Enquanto o primeiro apresenta estrutura espacial, que é diretamente relacionado com a concepção da meta-heurística, o segundo necessita de uma modelagem diferente para que o algoritmo seja aplicado com sucesso. Os algoritmos apresentados também incluem a utilização de busca local para melhora da solução. Além disto, resultados de experimentos de várias aplicações são reportadas para ambos os problemas.

Palavras-chave: Otimização por colônia de formigas, Problema do caixeiro viajante, Problema de cobertura de conjunto.

Abstract: *This chapter presents theoretical foundation and applications of the Ant Colony Optimization metaheuristic. We present algorithms of applications of this metaheuristic for the resolution of the Traveling Salesman Problem and the Set Covering Problem. While the former presents spatial structure, which is directly related to the design of the metaheuristic, the latter requires a different modeling in order to apply successfully the algorithm. The algorithms presented also include the use of local search procedure to improve the solution. Besides that, results of experiments of several applications are reported for both problems.*

Keywords: *Ant colony optimization, Traveling salesman problem, Set covering problem.*

1. Introdução

Algoritmos exatos para resolução de problemas de otimização combinatória $\mathcal{NP}$-difíceis possuem complexidade superpolinomial, a menos que $\mathcal{P} = \mathcal{NP}$ (Cormen et al., 2009). Uma alternativa para resolver tais problemas em tempo polinomial é a utilização de algoritmos heurísticos, geralmente embasados em alguma meta-heurística.

Meta-heurística é definida como um conjunto de conceitos algorítmicos e de estruturas de dados genéricos para desenvolvimento e aplicação de algoritmos heurísticos à resolução satisfatória de problemas $\mathcal{NP}$-difíceis (Dorigo & Stützle, 2004). Em geral, tais algoritmos não garantem encontrar a solução ótima, porém, avaliações estatísticas mostram que eles tem alcançado com regularidade a finalidade de retornar uma solução de boa qualidade em tempo computacional aceitável.

A Otimização por Colônia de Formigas (do inglês *Ant Colony Optimization* - ACO) (Dorigo & Stützle, 2004) é uma meta-heurística bio-inspirada que surgiu da observação do comportamento das formigas reais sendo que, posteriormente, foram agregadas técnicas de busca local. Em função desta analogia com o comportamento natural, as primeiras aplicações desta meta-heurística foram em problemas de otimização que tinham alguma associação com espaço físico, como o problema do caixeiro viajante (PCV).

O PCV pode ser descrito como o problema de um vendedor que necessita sair de sua cidade, percorrer todas as cidades contidas em uma área geográfica e retornar a sua cidade de origem de tal maneira que o percurso total seja mínimo e cada cidade seja visitada uma única vez. O PCV pode ser representado como um grafo completo $G = (V, E)$, sendo V o conjunto de cidades e E o conjunto de arestas que conectam todas as cidades. A cada aresta (i, j) é atribuído um valor d_{ij} que representa a distância entre as cidades i e j.

Em contraste ao PCV, também há a aplicação de ACO ao problema de cobertura de conjunto (PCC), que não possui relação com espaço físico. O PCC pode ser definido por uma matriz binária $A = [a_{ij}]$ de ordem $m \times n$. A cada coluna de A está associado um custo não-negativo c_j. Considera-se que uma coluna j de A

*Autor para contato: mhmulati@gmail.com

cobre uma linha i se $a_{ij} = 1$. O objetivo é selecionar um subconjunto de colunas $s \subseteq \{1, \ldots, n\}$ minimizando a soma de seus custos, de modo que cada linha seja coberta por ao menos uma coluna.

O objetivo deste capítulo é apresentar a meta-heurística ACO aplicada ao PCV e ao PCC, de forma a facilitar sua compreensão por meio da aplicação ao PCV e de modo a enfatizar a aplicação ao PCC, pelo fato de ser um problema de otimização combinatória que não possui uma associação direta com espaço físico, que é o caso de certos problemas de otimização formulados matematicamente. Destaca-se as características construtiva e melhorativa de ACO quando comparada com alguns algoritmos heurísticos encontrados na literatura. Do ponto de vista construtivo, este capítulo enfatiza o relacionamento da meta-heurística ACO com algoritmos gulosos aplicáveis aos problemas em estudo.

2. Problemas de Otimização Combinatória

Considera-se o problema de otimização combinatória de minimização [1] dado por $P = (S, \Omega, f)$, onde: S é o conjunto de soluções candidatas; Ω é um conjunto de restrições; e f é a função objetivo que associa um custo $f(s)$ a cada solução candidata $s \in S$. O objetivo é encontrar uma solução $s^* \in S$ globalmente ótima, i.e., que seja factível (respeita todas as restrições em Ω) e tenha o menor custo dentre todas as soluções candidatas. O modelo do problema de otimização combinatória é utilizado para definir o modelo de feromônio dos algoritmos ACO.

2.1 Problema do caixeiro viajante

O PCV pode ser representado como um grafo completo $G = (V, E)$, sendo V o conjunto de vértices (cidades) e E o conjunto de arestas (que conectam todas as cidades). A cada aresta (i, j) é associado um valor d_{ij} que representa a distância entre as cidades i e j. O objetivo é encontrar o menor caminho Hamiltoniano do grafo, caminho este fechado que visita cada vértice do grafo uma única vez. Caso o problema não deva ser mapeado em um grafo completo, pode-se retirar arestas ou atribuir $d_{ij} = \infty$ para as arestas ausentes.

As variáveis de decisão são dadas por x_{ij}, $\forall i, j \in \{1, 2, \ldots, n\}$, de modo que $x_{ij} = 1$ indica que, deve-se visitar a cidade i e imediatamente depois deve-se visitar a cidade j, utilizando assim a aresta (i, j) na solução; caso contrário, $x_{ij} = 0$. Para evitar soluções indesejadas, pode-se considerar $d_{ii} = \infty$ para $i = 1, 2, \ldots, n$, ou ainda, excluir as variáveis x_{ii}. De acordo com Colin (2007), o PCV pode ser modelado como segue:

$$\min f(x) = \sum_{i=1}^{n} \sum_{j=1}^{n} d_{ij} \cdot x_{ij} \tag{1}$$

sujeito a

$$\sum_{i=1}^{n} x_{ij} = 1, \ j = 1, 2, \ldots, n \tag{2}$$

$$\sum_{j=1}^{n} x_{ij} = 1, \ i = 1, 2, \ldots, n \tag{3}$$

$$u_i - u_j + n \cdot x_{ij} \leqslant n - 1, \ i \neq j; i = 2, 3, \ldots, n; j = 2, 3, \ldots, n \tag{4}$$

$$x_{ij} \in 0, 1, u_j \geqslant 0, \ j = 1, 2, \ldots, n; i = 1, 2, \ldots, n \tag{5}$$

A Equação 1 apresenta a função objetivo. As Equações 2 e 3 restringem que em uma solução cada vértice deve ter exatamente uma entrada e uma saída. Solução contendo todos os vértices mas com sub-rotas desconexas são evitadas pelas restrições da Equação 4, onde as variáveis u_i e u_j são auxiliares e não tem um significado físico. A Equação 5 apresenta as restrições de que as variáveis de decisão dadas por x_{ij} sejam inteiras e binárias. O valor u_j também é restringido.

Uma aplicação prática importante do PCV é o agendamento de uma máquina de fazer furos em uma placa de circuito impresso (Reinelt, 1994), onde os furos a serem feitos são representados pelas cidades e o custo da viagem é o tempo que se leva para mover a cabeça de perfuração de um orifício para o próximo. Deste modo, o objetivo é minimizar o tempo gasto para realizar todos os furos planejados para uma placa.

[1] A conversão em um problema de maximização geralmente é simples.

2.2 Problema de cobertura de conjunto

O PCC pertence à categoria de problemas de subconjunto, nos quais a solução é dada por um subconjunto de itens disponíveis, sujeita às restrições específicas do problema.

O PCC pode ser definido por uma matriz de ordem $m \times n$ definida por $A = [a_{ij}]$, na qual cada um dos elementos pode ser 0 ou 1. Cada coluna da matriz A possui um custo não-negativo c_j associado. Considera-se que determinada coluna j cobre uma linha i se $a_{ij} = 1$. O objetivo a ser alcançado no PCC é escolher um subconjunto solução $s \subseteq \{1, \ldots, n\}$ de colunas com custo mínimo, de modo que todas as linhas sejam cobertas. Este problema pode ser formulado como indicado a seguir:

$$\min f(x) = \sum_{j=1}^{n} c_j \cdot x_j \tag{6}$$

sujeito a

$$\sum_{j=1}^{n} a_{ij} \cdot x_j \geq 1, \ i = 1..m \tag{7}$$

$$x_j \in \{0, 1\}, \ j = 1..n \tag{8}$$

onde as restrições da Equação 7 definem que cada linha precisa ser coberta por ao menos uma coluna, e pelas restrições da Equação 8 tem-se que cada variável de decisão x_j tem que assumir 0 ou 1. A função objetivo é apresentada na Equação 6.

Algumas importantes aplicações do PCC são: escalonamento de pessoal em trânsito urbano (Desrochers & Soumis, 1989), escalonamento de pessoal em aeronaves (Housos & Elmroth, 1997), localização de serviços emergenciais (Toregas et al., 1971), recuperação de informação (Al-Sultan et al., 1996) e compactação de conjuntos de teste (Flores et al., 1999). Nestes casos reais, é comum encontrar instâncias do problema com centenas de linhas e milhares de colunas.

Em algumas situações, pode ser usado pré-processamento para reduzir uma instância de PCC, como descrito em de Oliveira (1999).

2.3 Algoritmo guloso

Conforme destacado por Dorigo et al. (1996), a meta-heurístca ACO possui três características básicas: uso de algoritmo construtivo guloso, comportamento auto-catalítico [2] e computação distribuída.

Algoritmo guloso (*greedy algorithm*, em inglês), também conhecido como algoritmo míope, é um algoritmo construtivo usado em ACO para encontrar soluções viáveis para o problema investigado. Tal algoritmo constrói iterativamente uma solução para um problema de otimização combinatória P a partir de uma solução vazia. A cada iteração o algoritmo escolhe um componente para compor a solução com base em uma "função gulosa" (também chamada de heurística gulosa). Uma função gulosa é uma função matemática que define heuristicamente a importância de um componente para entrar na solução a ser construída. A cada iteração, dentre os componentes candidatos, o componente com o maior valor da função gulosa é adicionado à solução. Empates podem ser resolvidos de maneira arbitrária. O algoritmo termina quando uma solução factível é construída. Nota-se, portanto, que a qualidade da solução construída depende de uma boa função gulosa.

Geralmente um algoritmo guloso é especializado para cada problema P. Há casos em que a ordem como os componentes são adicionados na solução é relevante, como para o PCV, pois a solução representa uma sequência de vértices (cidades). Porém, há casos em que a ordem não tem importância, como é o caso do PCC, pois a solução é um conjunto (não existe uma relação de ordem entre os componentes).

Um algoritmo baseado em ACO constrói as soluções utilizando duas informações numéricas (Dorigo et al., 1996): o valor da função gulosa e a quantidade de feromônio depositado pelas formigas. A escolha de cada componente para compor a solução é uma decisão tomada por cada formiga artificial baseada num valor probabilístico que combina estas duas informações numéricas.

Dentro do contexto dos algoritmos ACO esta função gulosa assume uma nova denominação chamada de "informação heurística" (algumas vezes denominada de visibilidade [3] da formiga). Para o PCV, a informação heurística para selecionar uma aresta (i, j), denotada por η_{ij}, pode-se basear, por exemplo, na ideia do clássico algoritmo vizinho mais próximo, que utiliza a função gulosa definida pela Equação 9:

$$\eta_{ij} = \frac{1}{d_{ij}} \tag{9}$$

[2] Um processo auto-catalítico (ou resposta positiva) é um processo de auto-reforço que ocorre com a utilização do feromônio depositado pelas formigas.

[3] Esta visibilidade é uma capacidade dada às formigas artificias de "enxergar" os elementos que compõem a solução, fazendo uma analogia ao percurso realizado pela formigas.

No caso do PCC, Chvàtal (1979) apresenta várias propostas como função gulosa, das quais destaca-se a função definida pela Equação 10, denotada por η_j, usada para selecionar uma coluna j para entrar na solução.

$$\eta_j = \frac{card_j(s)}{c_j} \tag{10}$$

onde $card_j(s)$ é o número de linhas que são cobertas pela coluna j mas não cobertas por nenhuma outra coluna na solução parcial s em construção, e c_j é o custo da coluna j.

Nota-se, ainda, que a informação heurística para o PCV é estática, pois ela não se modifica conforme a solução vai sendo construída. Já no caso do PCC a informação heurística é dinâmica, pois conforme a solução vai sendo construída o valor de $card_j(s)$ pode ser modificado em função das colunas já adicionadas à solução.

3. Otimização por Colônia de Formigas

A meta-heurística ACO foi introduzida por Colorni et al. (1992) com o *Ant System* (AS), porém, ainda sem fazer uso de algoritmos de busca local (Colorni et al., 1992; Dorigo, 1992; Dorigo et al., 1996). AS foi inspirado no comportamento de formigas reais – na busca de alimento para sua colônia – criado para ser utilizado na busca de soluções para problemas de otimização combinatória. Após a sua criação muitas variações foram elaboradas como nos casos de *MAX-MIN Ant System* (MMAS) (Stützle & Hoos, 1998), *Ant Colony System* (ACS) (Dorigo & Gambardella, 1997b), *ANTS* (Maniezzo, 1999), *MMAS-ACS-Hybrid* (Stützle & Hoos, 1997), *HyperCube* (Blum et al., 2001), *BeamACO* (Blum, 2005), dentre outros.

3.1 Inspiração e surgimento

Esta meta-heurística surgiu da observação do comportamento das formigas reais, ou seja, do estudo das formigas para entender como animais com pouca visão, como as formigas, poderiam conseguir uma rota mais curta partindo da colônia até uma fonte de comida. Foi descoberto que a comunicação entre as formigas que caminhavam pela trilha ocorria por meio de uma substância química, denominada feromônio, depositada por elas próprias. Enquanto as formigas caminham por uma trilha, inicialmente de forma aleatória, elas depositam uma certa quantidade de feromônio no solo. Assim, as próximas formigas tomam a decisão de seguir um caminho com probabilidade proporcional à quantidade feromônio depositada anteriormente. Ao decidir seguir um caminho com a presença da substância, ocorre então um reforço do caminho com o seu próprio feromônio. Este comportamento é denominado de auto-catalítico por ser um processo que reforça a si mesmo. O feromônio depositado tende a evaporar com o tempo, então quanto maior é a concentração de formigas passando pelo mesmo lugar, mais atrativo ele se torna para as próximas formigas. A Figura 1 ilustra um experimento com formigas reais.

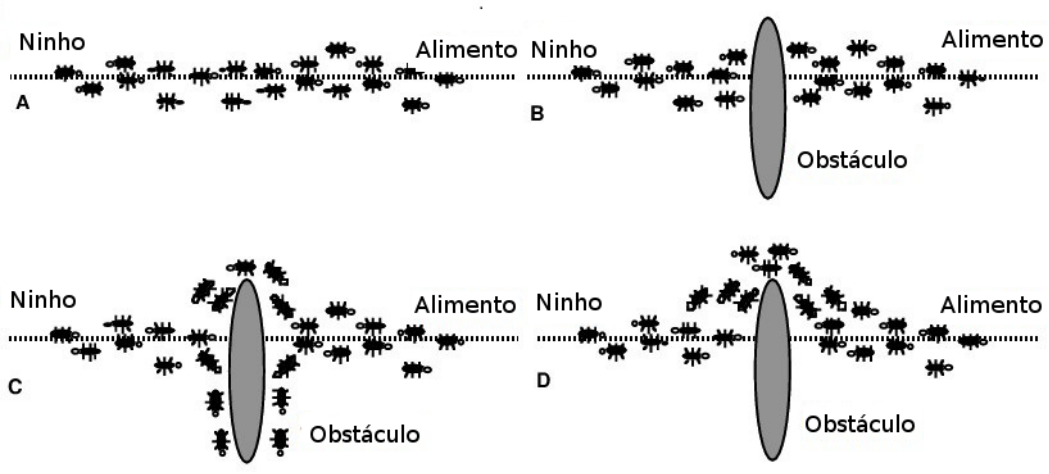

Figura 1. Ilustração do experimento com formigas. (A) As formigas seguem o caminho do ninho até o alimento. (B) Um obstáculo é colocado no caminho. (C) As formigas iniciam o desvio. (D) O caminho com maior frequência de formigas ocorre no caminho mais curto.

Desta forma, as formigas conseguem obter um bom caminho entre dois pontos. Na primeira decisão após a inserção do obstáculo, as quantidades de formigas a escolher o caminho mais curto e o caminho mais longo devem ser aproximadamente a mesma, de modo que elas escolhem o caminho com base no feromônio encontrado, sendo que as probabilidades de ambos os lados devem ter valores próximos um do outro. Porém, na segunda decisão, as formigas que percorreram o caminho mais curto já estão voltando, depositando ainda

mais feromônio, enquanto as que foram pelo caminho mais longo ainda estão completando a primeira transição. Desta forma, as formigas tendem a seguir caminho mais curto.

Com base no comportamento das formigas reais e com o objetivo de reproduzir tal comportamento, a Figura 2 introduz um esquema do ponto de vista das formigas artificias. Sendo que, t representa o tempo do sistema, d é a distância do caminho, A e E são os pontos extremos que as formigas devem atingir, e H e C são pontos intermediários, pelos quais as formigas devem decidir qual caminho tomar. O interessante é que escolham o mais curto, no caso, passando por C. De acordo com esta ilustração, no tempo $t = 0$ temos que metade do número de formigas (15) decidem ir do ponto B para o ponto C e a outra metade de B para H, considerando que ainda não há feromônio depositado no caminho. No tempo $t = 1$ considera-se que ocorreu depósito de feromônio pelas formigas no tempo $t = 0$. Porém, a quantidade de feromônio no caminho B, H, D é menor que no caminho B, C, D porque o primeiro caminho tem uma distância maior, portanto, as formigas consumiram mais tempo para percorrer, consequentemente houve maior evaporação do feromônio. Com a menor quantidade de feromônio no caminho B, H, D, há uma tendência de um menor número de formigas (no caso apenas 10) seguir tal caminho em relação a segunda alternativa. Desta forma, numa segunda iteração, $t = 1$, há uma maior concentração de formigas no caminho mais curto entre A e B.

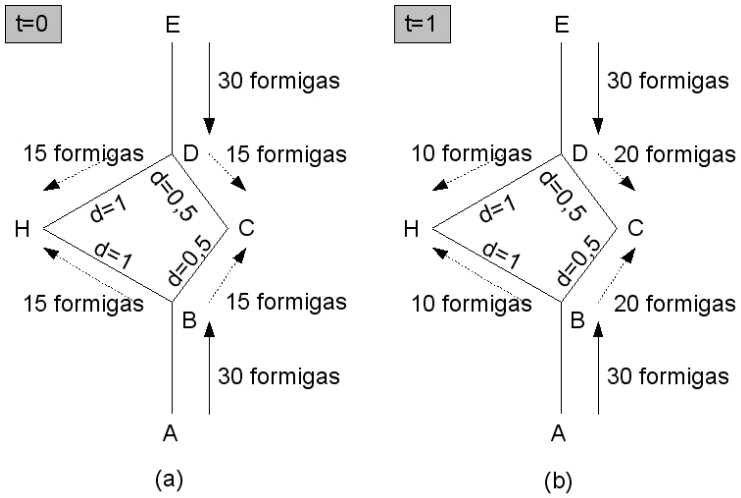

Figura 2. Exemplo de movimento das formigas artificiais.

A partir desta introdução nota-se que o tempo é discretizado em relação ao ambiente das formigas reais. Além disto, outras artificialidades são introduzidas em algoritmos ACO. As formigas artificiais são dotadas de boa capacidade de enxergar, diferente das formigas reais. Além do uso do feromônio já depositado (que será um valor numérico), na construção da solução (análoga ao processo de busca de alimento a partir do formigueiro) as formigas artificiais podem também se guiar pela informação heurística do caminho. No caso do PCV esta informação heurística é inversamente proporcional a distância entre dois vértices (Equação 9).

Outra artificialidade é o fato das formigas serem capazes de memorizar o caminho (solução) construído. Somente depois da construção da solução é que geralmente ocorre o depósito de feromônio pelas formigas artificiais com base em sua memória. Assim, como sugerido por Dorigo & Caro (1999) e Dorigo & Stützle (2004), a estrutura geral da meta-heurística ACO pode ser resumida pelo pseudo-código do Algoritmo 1, o qual possui três atividades (procedimentos): ConstruirSolucoesComFormigas(), AplicarBuscaLocal() e AtualizarFeromonio(). O procedimento principal da meta-heurística ACO é gerenciar o escalonamento dessas três atividades. Dorigo & Caro (1999) salientam que estas três atividades não possuem uma forma rígida de como são escalonadas e sincronizadas. Isto significa que estas atividades podem ser executadas de maneira paralela e independente, inclusive com sincronismo.

Algoritmo 1 Meta-heurística ACO.

MetaheuristicaACO()

```
1  Inicializar parâmetros;
2  Inicializar trilhas de feromônio;
3  while ( não alcançou a condição de parada ) do
4        ConstruirSolucoesComFormigas();
5        AplicarBuscaLocal();     // opcional
6        AtualizarFeromonio();
```

Os três procedimentos (atividades) são discutidos com mais profundidade nas seções de aplicações. O procedimento ConstruirSolucoesComFormigas() é essencialmente um algoritmo construtivo (guloso) aleatorizado que utiliza o feromônio e a informação heurística de forma combinada. Neste procedimento é explorada ideia de algoritmo guloso discutido em seção anterior. O feromônio é atualizado pelo procedimento AtualizarFeromonio(), podendo incluir o depósito e evaporação de feromônio. O procedimento AplicarBuscaLocal() tem a função de aplicar alguma busca local para melhorar uma solução, ou mais soluções, construída(s) por uma ou mais formigas. Apesar de ser opcional, o seu uso tem sido cada vez mais frequente. A condição de parada do laço de repetição pode ser baseada num número fixo de iterações e/ou convergência das soluções, além de outros critérios.

O feromônio e a informação heurística são valores que influenciam probabilisticamente a decisão da formiga. Todo o processo resulta em um sistema que exibe comportamento auto-catalítico: as formigas vão reforçando o feromônio das melhores soluções fomentando a convergência para a solução ótima.

ACO pode ser considerada simultaneamente uma meta-heurística com característica construtiva e melhorativa. A característica construtiva advêm do fato de usar a informação heurística tipicamente utilizada em algoritmo (construtivo) guloso. Além disto, este processo construtivo é repetido, porém, cada vez mais influenciado pelo feromônio introduzido em iterações anteriores com o objetivo de construir soluções cada vez melhores (que corresponde à característica melhorativa).

Aspectos importantes na aplicação de algoritmos ACO para um problema (Dorigo & Stützle, 2004) são: o grafo de construção, sobre o qual as formigas caminham; as restrições; a definição das trilhas de feromônio; a definição da informação heurística; a probabilidade de escolha; a construção da solução; a atualização de feromônio; e a busca local.

4. Aplicações de ACO

Na literatura podem ser encontradas diversas aplicações de algoritmos ACO em problemas de otimização combinatória, sendo alguns deles: problema do caixeiro viajante (PCV) (Dorigo et al., 1996), problema generalizado de atribuição (PGA) (Ramalhinho-Lourenço & Serra, 1998), problema quadrático de alocação (PQA) (Maniezzo & Colorni, 1999), problema de cobertura de conjunto (PCC) (Hadji et al., 2000), problema da mochila multidimensional (Alaya et al., 2004), problemas de clique máximo em grafos (Solnon & Fenet, 2006) problema de roteamento de pacotes em redes de computadores (Dorigo & Stützle, 2004), escalonamento de tripulações (Deng & Lin, 2011) e problemas de roteirização de veículos (Wade & Salhi, 2004).

Esta seção mostra a aplicação do ACO ao problema do caixeiro viajante e ao problema de cobertura de conjunto, com o objetivo de ilustrar duas aplicações distintas da meta-heurística ACO.

4.1 Algoritmo ACO para o problema do caixeiro viajante

Em Dorigo & Stützle (2004) é realizada uma discussão sobre a aplicação de algoritmos ACO para o PCV. Nesta seção é apresentada uma proposta de implementação das particularidades dos procedimentos apresentados no Algoritmo 1.

No PCV, o grafo de construção – espaço utilizado pelas formigas artificiais – e o grafo que representa o problema são idênticos. Em cada passo de construção uma formiga deve visitar um vértice (cidade) adjacente entre aqueles que ainda não foram visitados e ao final deve retornar ao vértice inicial. Todos os vértices devem ser visitados uma única vez.

Neste problema, os resíduos de feromônio indicam a desejabilidade de visitar uma determinada cidade i, partindo da cidade j, sendo denotados por τ_{ij}. Além disto, a informação heurística η_{ij} é tipicamente inversamente proporcional à distância entre i e j, como apresentada na Equação 9, inclusive sendo bastante utilizada em algoritmos gulosos. A informação de escolha é denotada por ${\tau_{ij}}^{\alpha} \cdot {\eta_{ij}}^{\beta}$, sendo α e β parâmetros que indicam a importância do feromônio e da informação heurística, respectivamente.

Cada formiga é inicializada em uma cidade. A cidade pode ser escolhida aleatoriamente ou, caso a quantidade de formigas seja maior que a quantidade de cidades, serem distribuídas de modo que cada cidade possua ao menos uma formiga. Todas as formigas constroem sua solução executando seus passos de construção. A cada passo de uma formiga ela seleciona uma cidade vizinha da atual ainda não visitada para fazer parte da solução, seleção esta que ocorre levando em conta a informação de escolha. A construção da solução termina quando todas as cidades foram visitadas pela formiga. Para selecionar uma nova cidade a ser adicionada a sua solução, a formiga seleciona tal cidade de acordo com a probabilidade de escolha p dada pela Equação 11:

$$p_{ij}^{k} = \begin{cases} \dfrac{\tau_{ij}^{\alpha} \cdot \eta_{ij}^{\beta}}{\sum_{h \in N(i,s_k)} \tau_{ih}^{\alpha} \cdot \eta_{ih}^{\beta}} & \text{se } j \in N(i,s_k) \\ 0 & \text{se } j \notin N(i,s_k) \end{cases} \tag{11}$$

onde $N(i, s_k)$ é o conjunto de cidades candidatas (novas) que podem ser selecionadas a partir da cidade i com a solução parcial s_k já construída pela formiga k, e s_k é definida como uma estrutura vetorial de tamanho n, sendo n o número de cidades (vértices). Assim, com este critério probabilístico de seleção de candidatos é possível construir um algoritmo guloso aleatorizado, em que cada vértice selecionado compõe uma solução (rota).

Um aspecto que chama atenção para um algoritmo ACO é a quantidade de parâmetros. Além dos parâmetros α e β, tem-se ainda:

- ρ: taxa de evaporação de feromônio;
- NI_{max}: o número máximo de iterações do algoritmo;
- τ_0: a quantidade de feromônio a ser atribuída para as arestas na inicialização do algoritmo;
- m: o número de formigas artificias utilizadas para este algoritmo.

Encontrar uma atribuição de valores para estes parâmetros é sempre um desafio para se projetar um algoritmo baseado em ACO. Por exemplo, em Dorigo et al. (1996) são sugeridos os seguintes valores de parâmetros: $m = n$, $\alpha = 1$, $\beta = 0,5$, $\rho = 0,5$ e $NI_{max} = 5000$. Sugere-se usar $\tau_0 = 0,05$.

O procedimento ConstruirSolucoesComFormigas() (usado pelo Algoritmo 1) realiza a tarefa de construção de soluções e é representado pelo Algoritmo 2.

Algoritmo 2 Construção de soluções com formigas para o PCV.

ConstruirSolucoesComFormigas()

```
1  for  cada aresta (i, j)  do
2       τ_ij = τ_0;            // Feromônio inicial em cada aresta
3  for  k = 1 to m  do
4       for  j = 1 to n  do
5            s_k[j] = 0;            // esvaziar a rota da formiga k
6  for  k = 1 to m  do
7       while  Tiver cidade que ainda não está em s_k  do
8            Selecione a cidade j com probabilidade p_ij^k da Equação 11;
9            Inserir j na solução s_k;
10           Mover a formiga k para a cidade j;
11 for  k = 1 to m  do
12      Mover a formiga k para a cidade s_k[1];
13      Calcular o custo L_k da rota obtida pela formiga k;
```

Após cada formiga ter construído sua solução, ocorre a atualização de feromônio por meio da evaporação e do depósito de feromônio. A atualização é dada pela Equação 12:

$$\tau_{ij} = (1 - \rho)\tau_{ij} + \sum_{k=1}^{m} \Delta\tau_{ij}^{k}, \quad \forall i, j \in E \tag{12}$$

onde $\Delta\tau_{ij}^{k}$ é descrito pela Equação 13:

$$\Delta\tau_{ij}^{k} = \begin{cases} \dfrac{1}{L^k} & \text{se a aresta } (i,j) \text{ pertence a solução } s_k \\ 0 & \text{caso contrário} \end{cases} \tag{13}$$

sendo que L^k representa o custo da solução s_k, que é a soma dos custos de todas as arestas que fazem parte da rota construída pela formiga k. Assim, quanto melhor a solução, mais feromônio será depositado em suas arestas. Neste esquema, todas as formigas depositam feromônio. Com isto, o procedimento AtualizarFeromonio() (usado pelo Algoritmo 1) se resume na aplicação da Equação 12.

A inicialização é uma operação simples que estabelece as condições iniciais do algoritmo. Neste caso, é necessário realizar:

- $NI = 0$, sendo que NI define o número de iterações do algoritmo;
- Alocar arbitrariamente cada uma das m formigas nos n vértices (cidades), ou seja, $s_k[1]$ receberá a primeira cidade da rota para a formiga k.

Este algoritmo adota dois critérios de parada:

- $NI \geq NI_{max}$, que impõe um número máximo de iterações para o algoritmo;
- "Estagnação": a estagnação das soluções ocorre quando todas (ou quase todas) as soluções são equivalentes, ou seja, todas (ou quase todas) as formigas passaram a fazer a mesma rota. Quanto a estagnação ocorre é um sinal que o algoritmo deve parar porque não encontrará mais soluções diferentes. Esta estagnação pode estar relacionada com os parâmetros utilizados pelo algoritmo.

Embora a busca local seja opcional, ela é bastante recomendada por ser responsável por melhorar a qualidade da solução produzida pelas formigas. Exemplos bem conhecidos na literatura são 2-opt (Croes, 1958), 3-opt (Lin, 1965) e Lin-Kerninghan (Lin & Kernighan, 1973).

4.2 Algoritmo ACO para o problema de cobertura de conjunto

Em Dorigo & Stützle (2004) são revisitadas duas aplicações do algoritmo *Ant-System* (AS) ao PCC: o algoritmo AS-LM, com ideias de Leguizamon & Michalewicz (1999), e o algoritmo AS-HRTB, de Hadji et al. (2000). O segundo trabalho teve resultados estendidos pelo algoritmo *AntsLS* em Rahoual et al. (2002), além da inclusão de uma versão paralela. Constam nos dois últimos trabalhos a utilização da busca local JB, embasada em esquema de Jacobs & Brusco (1995). O trabalho de Lessing et al. (2004) faz comparativos de aplicação ao PCC dos algoritmos MMAS, ACS, *MMAS-ACS-Hybrid* e ANTS, com e sem a busca local *3-FLIP*, baseada em Yagiura et al. (2006), avaliando também diferentes tipos da informação heurística.

Em Mulati & Constantino (2011) é proposto e investigado o algoritmo *Ant-Line* (AL), que constrói soluções com base na seleção de linhas da instância do PCC, característica chamada "orientação a linha". AL também utiliza a busca local JB. Orientação a linha também foi trabalhada em Ren et al. (2008) com o *Ant-Cover* (AC), comparando seus resultados com o MMAS, com e sem busca local. Em Ren et al. (2010) o AC com busca local, esta embasada na eliminação de colunas redundantes, é comparado com outras meta-heurísticas e com abordagem ACO de Crawford & Castro (2006), que inclui etapa de pós-processamento.

Estendendo o algoritmo heurístico genérico ACO apresentado no Algoritmo 1 e utilizando os conceitos introduzidos na aplicação de ACO para PCV apresentados na Seção 4.1, segue aprofundamento na aplicação de algoritmos ACO ao PCC.

4.2.1 AS para o PCC

No grafo de construção, as colunas são representadas pelos componentes do conjunto de vértices C, podendo existir um componente extra que não está relacionado com nenhuma das colunas do PCC. Em geral, o grafo de construção é completo, indicando que após a inserção de uma coluna, não há uma restrição rígida de colunas que não podem ser inseridas na sequencia. O componente extra geralmente serve de ponto de partida para as formigas, que normalmente não podem retornar a ele durante a construção de uma solução.

O algoritmo deve gerir como as restrições do problema serão tratadas, sendo que no PCC não faz sentido construir soluções com colunas repetidas. Deste modo, não é permitido à formiga adicionar à solução um componente que já se encontra nesta. Tal mecanismo é elaborado através do conjunto $N(s_k)$, que consiste dos componentes candidatos de serem inseridos na solução parcial s_k pertencente à formiga k. Assim, a formiga k deve procurar novos componentes para sua solução em $N(s_k)$, este que contém todos os componentes de C (exceto o componente extra) que não estão na solução s_k.

Os resíduos de feromônio são associados com os componentes, desta forma o feromônio τ_j associado com o componente j mede a desejabilidade de se incluir a coluna j na solução. No PCC enfatiza-se a utilização da informação heurística dinâmica custo de cobertura, dada por η_j, como apresentada na Equação 10, onde deve-se considerar que $s = s_k$ antes da utilização da regra (e também antes de calcular $card_j(s)$). A informação de escolha é expressa por ${\tau_j}^{\alpha} \cdot {\eta_j}^{\beta}$ usando valores de um componente (coluna) j, já com a utilização dos parâmetros α e β, que indicam a importância do feromônio e da informação heurística, respectivamente.

O Algoritmo 3 apresenta o procedimento ConstruirSolucoesComFormigas(), que é chamada em cada iteração do Algoritmo 1. Em tal procedimento, cada formiga constrói sua solução.

O procedimento AplicarPassoDeConstrucao(s_k) é responsável por selecionar um componente j e adicioná-lo em s_k com base na probabilidade de escolha p, definida pela Equação 14:

$$p_j^k = \begin{cases} \dfrac{\tau_j^{\alpha} \cdot \eta_j^{\beta}}{\sum_{h \in N(s_k)} \tau_h^{\alpha} \cdot \eta_h^{\beta}} & se\ j \in N(s_k) \\ 0 & se\ j \notin N(s_k) \end{cases} \tag{14}$$

O conjunto $N(s_k)$ contém os componentes candidatos, que são aqueles ainda não selecionados para a solução s_k da formiga k e que cobrem ao menos uma linha ainda não coberta. No Algoritmo 1, a etapa de busca

Algoritmo 3 Construção de soluções com formigas para o PCC.

```
ConstruirSolucoesComFormigas()
1  for  k = 1 to m  do   // Cada formiga constrói sua solução
2       s_k = {};
3       while  (s_k não está completa)  do
4            AplicarPassoDeConstrucao(s_k);// Seleciona componente
5       AplicarBuscaLocal(s_k);              // Opcional
6       EliminarColunasRedundantes(s_k);    // Opcional
```

local aparece em bloco após todas as formigas terem construído suas soluções, ao passo que no Algoritmo 3 a busca local (Linha 5) é aplicada imediatamente após cada formiga ter construído sua solução. Porém, ressalta-se que a ideia do segundo se encaixa no esquema clássico apresentado pelo primeiro considerando que (no primeiro) após todas as formigas construírem suas soluções, a busca local é aplicada a cada uma das soluções obtidas pelas formigas. Isto também pode acontecer no Algoritmo 3, porém não se aguarda que todas as soluções sejam construídas para então aplicar a busca local. Em seguida, pode-se realizar a eliminação de colunas redundantes que possam existir na solução (Linha 6), i.e., aplicar procedimento para remover colunas cujas linhas cobertas já são cobertas por outras colunas, sendo que, removendo-se a referida coluna, implicará apenas em diminuição do custo sem prejuízo da cobertura de linhas da solução. A busca local e a eliminação de colunas redundantes são procedimentos opcionais.

O *Ant-Line* (Mulati & Constantino, 2011) segue esquema próximo do apresentado no Algoritmo 3 e os algoritmos apresentados em Lessing et al. (2004) seguem esquema semelhante, porém as linhas 5 e 6 são invertidas. O AS-LM realiza o procedimento de seleção descrito, porém sem busca local e sem eliminação de colunas redundantes. O AS-HRTB realiza o processo semelhante, porém ao invés de começar com uma solução vazia, inicia com uma solução parcial contendo componentes selecionados aleatoriamente, a fim de aumentar a diversificação da solução. Outra diferença é que o AS-HRTB possui uma remoção de componentes redundantes da solução ao fim de cada passo de construção. O AS-HRTB aplica busca local somente ao fim do processo de construção de soluções.

Na atualização de feromônio, primeiro deve ocorrer a evaporação do feromônio para que então ocorra o depósito do mesmo. A evaporação é dada pela Equação 15 e o depósito é mostrado na Equação 16:

$$\tau_j = (1-\rho)\cdot\tau_j,\ \forall j \in C \tag{15}$$

$$\tau_j = \tau_j + \sum_{k=1}^{m} \Delta\tau_j^k \tag{16}$$

A quantidade de formigas é dada por m, que normalmente é um parâmetro do algoritmo. Neste esquema, todas as formigas depositam feromônio. No AS-HRTB, o valor da a ser depositado é dado por $\Delta\tau_j^k = 1/f(s_k)$, onde $f(s_k)$ é o valor da função objetivo da solução s_k da formiga k se o componente j é um elemento de s_k, e tem valor 0 caso contrário. O algoritmo AS-LM utiliza regra semelhante, com a diferença de que o feromônio depositado é multiplicado pela soma dos custos de todas as colunas na definição do problema.

Com respeito à busca local, o AS-HRTB aplica a busca local JB à melhor solução construída na última iteração do algoritmo. A JB efetua um perturbação da solução favorecendo a inclusão de colunas com bom custo-benefício.

O AS-HRTB utiliza como critério de parada, em Hadji et al. (2000), uma quantidade fixa de iterações (NI_{max}) definida como parâmetro, que foi definida com o valor 15. Ainda sobre os parâmetros, a quantidade de formigas (m) também recebeu valor 15. Em parte dos experimentos $\alpha = 1$, $\beta = 5$ e $\rho = 0,5$. Uma forma de se inicializar os resíduos de feromônio é fazendo $\tau_0 = 1$, como feito em Mulati & Constantino (2011).

4.2.2 O algoritmo *Ant-line*

O algoritmo *Ant-line* – AL com aplicação no PCC, proposto em Mulati & Constantino (2011), segue as linhas gerais da meta-heurística ACO apresentada no Algoritmo 1. Seu funcionamento se assemelha ao do AS, porém com significavas alterações. A estrutura da construção de solução é a mesma do Algoritmo 3, mas com diferenças fundamentais na seleção do próximo componente da solução efetuada no passo de construção.

Assim, na aplicação do passo de construção, o AL não leva em consideração apenas a informação de escolha. Cada formiga k é um método construtivo que, a cada passo de construção deve: (1) selecionar aleatoriamente uma linha e que ainda não é coberta por nenhuma coluna na solução s_k e (2) escolher uma coluna para cobrir esta linha utilizando uma regra de decisão determinística baseada na informação de escolha.

No estágio (1), a decisão seleciona uma linha e com probabilidade dada pela distribuição uniforme, conforme a Equação 17:

$$p_e^k = \begin{cases} \frac{1}{|M \setminus R(s_k)|} & \text{se } e \notin R(s_k) \\ 0 & \text{se } e \in R(s_k) \end{cases} \quad \forall e \in M \tag{17}$$

onde $R(s_k)$ é o conjunto de todas as linhas cobertas pelos componentes que já estão em s_k, e M é o conjunto que contém todas as linhas.

Pela seleção da linha e, o algoritmo define um conjunto de componentes candidatos que é dado pela Equação 18:

$$N(e, s_k) = \{j | (j \notin s_k) \wedge (e \notin R(s_k)) \wedge (a_{ej} = 1)\}, \forall j \in C \tag{18}$$

O estágio (2) deterministicamente seleciona um componente do conjunto de componentes candidatos $N(e, s_k)$. Isto é feito de acordo com a Equação 19:

$$j = argmax_{h \in N(e,s_k)}\{\tau_h^\alpha \cdot \eta_h^\beta\} \tag{19}$$

A informação heurística utilizada é a custo de cobertura apresentada na Equação 10, onde é necessário considerar as mesmas observações feitas para o AS. O passo de construção de solução é ilustrada no Algoritmo 4.

Algoritmo 4 Passo de construção de formiga do *Ant-Line*.

AplicarPassoDeConstrucao(s_k)

1 e = Selecionar aleatoriamente uma linha que ainda não é coberta por nenhuma coluna em s_k; // Como na Equação 17

2 $j = argmax_{h \in N(e,s_k)}\{\tau_h^\alpha \cdot \eta_h^\beta\}$, onde $N(e,\ s_k)$ contém todas as colunas que cobrem a linha e, exceto aquelas em s_k; // Como nas Equações 19 e 18

3 $s_k = s_k \cup \{j\}$;

Quando todas as formigas tiverem construído suas soluções, os resíduos de feromônio são atualizados pela evaporação e depósito. A evaporação é feita de acordo com a Equação 15. Depois disto, o feromônio é depositado. Para embasar este procedimento, é necessário fazer a solução s receber s' (melhor solução da iteração) ou s^* (melhor solução da execução da tentativa), que é feito de forma que inicialmente s' reforça mais os resíduos do que s^*. Uma mudança gradual nesta frequência é feita com base na quantidade máxima de iterações do algoritmo que foi recebida como parâmetro. Assim, o depósito de feromônio é feito por uma única formiga de acordo com a Equação 20:

$$\tau_j = \tau_j + \left(\frac{f(s^*)}{f(s)}\right)^y, \forall j \in s \tag{20}$$

onde y é um parâmetro que regula o valor a ser depositado nos componentes da solução.

A busca local utilizada é a JB. O algoritmo também pode realizar a reinicialização de feromônio caso s^* não tenha melhoria dentro de uma quantidade parametrizada de iterações. Mais detalhes do funcionamento do AL podem ser encontrados em Mulati & Constantino (2011).

5. Resultados

A literatura apresenta diversas aplicações de algoritmos ACO a problemas de otimização combinatória (Dorigo & Stützle, 2004). As aplicações e experimentos variam em diversos aspectos. As aplicações diferem entre si desde pequenas decisões nos projetos de algoritmo até mudanças substanciais no modo como as formigas trabalham, além de terem parâmetros diferentes. As instâncias utilizadas diferem em tamanho e características, por exemplo. Os equipamentos de *hardware* utilizados em cada experimento normalmente possuem características distintas. Dadas estas situações, a presente Seção apresenta alguns resultados de ACO aplicado a PCV e ao PCC de modo a ilustrar tais aplicabilidades e analisar algumas qualidades de soluções, sem se prender ao tempo de processamento nem a comparações rígidas entre abordagens diferentes.

5.1 ACO para PCV

Uma experiência relacionada com a aplicação da ACO ao PCV é reportada em Dorigo & Gambardella (1997a) utilizando instâncias de PCV. As instâncias usadas são Oliver30 (Oliver et al., 1987), Eil50 e Eil75 (Eilon et al., 1971) e Kro100 da base de dados TSPLIB[4]. Os resultados são comparados com outras meta-heurísticas.

[4] Disponível em http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/, acessado em abril de 2012.

Os números que compõem os nomes das instâncias correspondem ao seus respectivos números de vértices (cidades). Estes resultados são resumidos na Tabela 1.

Tabela 1. Comparação do ACO com algoritmos genéticos (AG) e *simulated annealing* (SA) para o PCV reportada por Dorigo & Gambardella (1997a). A última coluna representa o valor na melhor solução conhecida para a instância.

Nome do problema	ACO	AG	SA	Solução Ótima
Oliver30	**420**	421	424	420
Eil50	**425**	428	443	425
Eil75	**535**	545	580	535
Kro100	**21282**	21761	-	21282

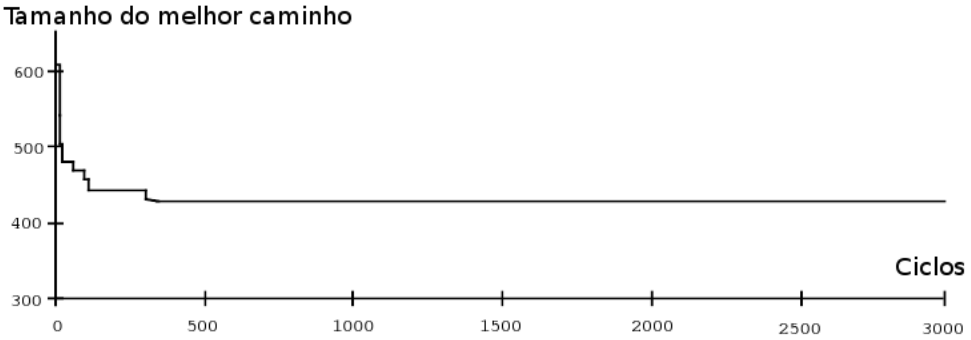

Figura 3. Evolução do custo do melhor caminho (Oliver30). Execução típica. Fonte: Dorigo et al. (1996)

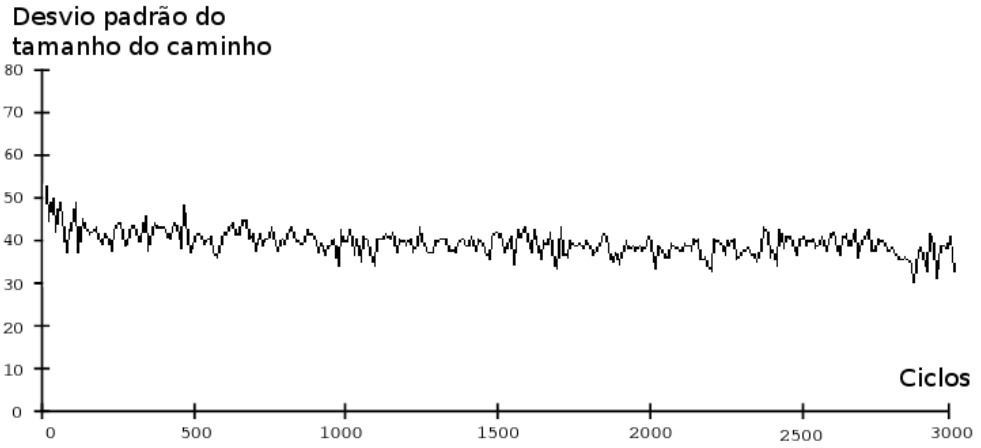

Figura 4. Evolução do desvio padrão dos custos de caminho da população (Oliver30). Execução típica. Fonte: Dorigo et al. (1996)

Um aspecto interessante pode ser observado nas Figuras 3 e 4, que mostram a execução do algoritmo para a instância Oliver30. Pela Figura 3 é possível ver que o melhor caminho evolui até o ponto em que se estabiliza.

A Figura 3 mostra que este ponto ocorre com menos de 500 ciclos (iterações). Porém, a Figura 4 mostra que o desvio padrão do custo das soluções encontradas pelas formigas continua variando, o que demonstra que o algoritmo continua pesquisando por novas soluções, mesmo após a estabilização da melhor solução encontrada mostrada na Figura 3. Este é um aspecto importante de um algoritmo ACO: que não entre em estagnação e tenha condições de escapar de ótimos locais.

5.2 ACO para PCC

Experimentos foram conduzidos em instâncias de PCC provenientes da *OR-Library* [5] (Beasley, 1990). Estas instâncias estão divididas nas classes de PCC (C.PCC): `4`, `5`, `6`, `a`, `b`, `c`, `d`, `nre`, `nrf`, `nrg` e `nrh`; além da classe `e`, que não é usada em todos os experimentos reportados. As classes `4` e `5` possuem 10 instâncias cada, ao passo que todas as demais possuem 5 instâncias. A classe `4` possui dimensão 200×1000 (linhas $\times$ colunas) e a classe `5` tem dimensão 200×2000, enquanto ambas possuem densidade de 2%. A classe `6` tem dimensão 200×1000 com densidade 5%. Nas classes `a`, `b`, `c` e `d`, as duas primeiras têm dimensão 300×3000 enquanto que as demais, 400×4000; com as densidades sendo 2%, 5%, 2% e 5%, respectivamente. Estas classes serão referidas em conjunto como `4-d`. A classe `nre` tem densidade 10% e `nrf` tem 20%, de modo que ambas têm dimensão 500×5000. De dimensão 1000×10000 são as classes `nrg` e `nrh`, com densidades 2% e 5%, respectivamente. As últimas quatro classes citadas serão referenciadas como `nre-nrh`. A classe `e` tem dimensão 50×500 com densidade de 20%.

Todos os experimentos foram conduzidos de forma que em cada execução de uma configuração de algoritmo e instância de problema foram realizadas dez tentativas semelhantes, variando-se os apenas as sequencias de números aleatórios utilizados. É importante o fato de que os algoritmos foram executados em diferentes hardwares, e assim, a análise tem como enfoque as qualidades das soluções obtidas. Com respeito a uma execução e em relação à solução ótima ou melhor conhecida (OMC), seguem designações que aparecem nos resultados: solução média em percentual de distância para a solução OMC (%) (SMd(%)), melhor solução (%) (MS(%)), e pior solução (%) (PS(%)). Aparecem também os tempos tempo médio da tentativa em segundos (s) (TMdT(s)) e tempo médio para encontrar a primeira melhor solução das tentativas (s) (TMdEP(s)).

Os resultados computacionais obtidos pelos algoritmos AS-LM e AS-HRTB (Hadji et al., 2000) são bons, porém não alcançam as melhores versões de algoritmos ACO. O mesmo vale para o *AntsLS*, de Rahoual et al. (2002), algoritmo cujos resultados são reportados na Tabela 2. Os tempos de execução do *AntsLS* variam de 50s a 460s para instâncias `4-d` e de 46s a 3000s para `nre-nrh`.

Tabela 2. Resultados de *AntsLS* (Rahoual et al., 2002).

C.PCC	**SMd**(%)	**MS**(%)
4 (10)	0,50	0,25
5 (10)	0,70	0,34
6 (5)	1,51	0,56
a (5)	0,98	0,26
b (5)	0,25	0,25
c (5)	1,03	0,36
d (5)	0,95	0,56
nre (5)	0,74	0,74
nrf (5)	1,97	1,54
nrg (5)	2,21	0,97
nrh (5)	1,94	0,98
Média	1,07	0,57

Tabela 3. Resultados de Lessing et al. (2004) e de AL (Mulati & Constantino, 2011), ambos com respectivas buscas locais.

Algoritmo	**#OMC**	**REL**
MMAS	685	699,6
ACS	679	699,5
MAH	684	699,6
ANTS	683	699,6
AL	435	696,6

Aplicações de vários algoritmos ACO para PCC são realizados em Lessing et al. (2004), e parte dos resultados referente à utilização com busca local e informação heurística custo de cobertura é mostrada na Tabela 3 (onde MAH refere-se a *MMAS-ACS-Hybrid*), que considera todas as instâncias apresentadas, incluindo as da classe `e`. O valor #OMC indica a quantidade de soluções ótimas ou melhores conhecidas encontradas pelos algoritmos. O valor REL, estipulado pelo trabalho originário, é denotado por: a soma das divisões dos valores ótimos ou melhores conhecidos pelos valores das soluções encontradas em cada tentativa. Desta maneira, ocorre a avaliação do comportamento dos algoritmos de forma geral, de modo que quanto mais perto de 700, melhor é o resultado obtido. O tempo de execução foi fixado em 100s para todos os testes de Lessing et al. (2004) apresentados. Calculou-se o valor para execuções similares do AL, o que é mostrado na última linha da Tabela. O tempo do AL não é fixo como o dos outros algoritmos, com o TMdT consumindo 1689,37s e o TMdEP com valor 22,04s. Assim AL é pouco pior em qualidade de solução e é pior que os outros em tempo de execução geral, mas o baixo TMdEP indica bom potencial para melhora geral do tempo.

Aplicações que utilizam a orientação a linha para o PCC ocorrem nos algoritmos ACO AC, de Ren et al. (2010) e AL, de Mulati & Constantino (2011). Os resultados de AL são ilustrados na Tabela 4. É possível

[5] Disponível em `http://people.brunel.ac.uk/~mastjjb/jeb/orlib/scpinfo.html`, acessado em abril de 2012.

fazer comparação do AL com o *AntsLS*. O segundo possui médias SMd=1,07% e MS=0,57%, enquanto que o primeiro tem SMd=0,53% e MS=0,12%, denotando melhor qualidade de solução para o AL.

Tabela 4. Resultados de AL com busca local (Mulati & Constantino, 2011).

C.PCC	TMdT(s)	TMdEP(s)	PS(%)	SM(%)	MS(%)
4 (10)	8,23	2,53	0,48	0,19	**0**
5 (10)	23,44	6,2	0,42	0,19	**0**
6 (5)	5,18	1,84	0,98	0,36	**0**
a (5)	221,78	43,66	0,75	0,38	0,08
b (5)	26,86	4,18	0,25	0,12	**0**
c (5)	401,18	71,62	0,63	0,36	0,17
d (5)	178,36	17,22	0,92	0,59	**0**
nre (5)	2610,98	175,26	1,41	0,5	**0**
nrf (5)	2861,82	166,54	2,97	1,21	**0**
nrg (5)	8640,4	2130,54	1,81	0,91	0,37
nrh (5)	8640,68	1299,18	2,29	1,67	0,97
Média	1819,28	280,54	1,06	0,53	0,12

O AC é configurado com tempo máximo baixo, variando de 2s a 20s, além de utilizar informação heurística mais elaborada, que é embasada em relaxação Lagrangeana que trabalha com método subgradiente (Fisher, 1981) para obter limite inferior para solução ótima pelo problema dual. Por sua vez, o AL utiliza vários critérios de parada, dentre eles o tempo máximo, que é configurado em valores relativamente elevados. Os tempos médios do AC são TMdT=14,65s e TMdEP=4,27s, ao passo que AL consome TMdT=1819,28s, TMdEP=280,54s. A grande diferença entre TMdEP e TMdT do AL indicam potencial de melhora do TMdT por ajuste dos critérios de parada. A média da qualidade de solução do AC fica em PS=0,33%, SMd=0,17% e MS=0,02%; enquanto que o AL tem valores próximos, que são PS=1,06%, SMd=0,53% e MS=0,12%.

6. Conclusões

A meta-heurística ACO tem sido aplicada com sucesso a vários problemas de otimização combinatória. Este capítulo apresentou a meta-heurística ACO, ilustrando sua estrutura básica e ilustrando sua aplicação em dois problemas clássicos de pesquisa operacional, sendo o problema do caixeiro viajante (PCV) e o problema de cobertura de conjunto (PCC).

O PCV e o PCC foram utilizados para exemplificar o uso de ACO por serem problemas clássicos e por apresentarem características bastante distintas. O primeiro explora o relacionamento espacial do problema em analogia com o comportamento das formigas reais, onde o grafo do problema é relacionado com o ambiente natural das formigas. Enquanto que o segundo caso explora um problema de otimização combinatória puramente matemático, em que não há relação com elementos espaciais que permita fazer analogia com o ambiente espacial das formigas reais.

Por outro lado, este estudo comparativo com dois tipos de problemas distintos possibilita demonstrar que a concepção de um algoritmo baseado em ACO para um problema de otimização não requer associação do problema com aspectos espaciais do ambiente das formigas reais.

Um aspecto fundamental, observado na concepção de algoritmo baseado em ACO, é a investigação de um algoritmo construtivo (guloso). Dorigo et al. (1996) observam que quando o parâmetro $\alpha = 0$ (Equação 11), o algoritmo ACO torna-se um clássico algoritmo guloso aleatorizado, e que, quando $\beta = 0$, tem-se um algoritmo baseado somente no feromônio, que geralmente converge rapidamente para um ótimo local. Esta observação salienta a importância de se ajustar estes dois parâmetros de forma que a função gulosa utilizada não seja ignorada. O uso de algoritmos gulosos também é observado na meta-heurística GRASP (*Greedy Randomized Adaptive Search Procedure*) (Resende et al., 1996). Portanto, a investigação de um algoritmo baseado em GRASP também pode ser uma fonte de informação importante para se projetar um algoritmo baseado em ACO.

Referências

Al-Sultan, K.S.; Hussain, M.F. & Nizami, J.S., A genetic algorithm for the set covering problem. *Journal of the Operational Research Society*, 47(5):702–709, 1996.

Alaya, I.; Solnon, C. & Ghedira, K., Ant algorithm for the multidimensional knapsack problem. In: *Proceedings of International Conference on Bioinspired Optimization Methods and their Applications*. p. 63–72, 2004.

Beasley, J.E., OR-Library: distributing test problems by electronic mail. *Journal of the Operational Research Society*, 41(11):1069–1072, 1990.

Blum, C., Beam-ACO – hybridizing ant colony optimization with beam search: an application to open shop scheduling. *Computers and Operations Research*, 32(6):1565–1591, 2005.

Blum, C.; Roli, A. & Dorigo, M., HC-ACO: the hyper-cube framework for ant colony optimization. In: *Proceedings of Metaheuristics International Conference*. p. 399–403, 2001.

Chvàtal, V., A greedy heuristic for the set-covering problem. *Mathematics of Operations Research*, 4(3):233–235, 1979.

Colin, E.C., *Pesquisa Operacional: 170 Aplicações em Estratégia, Finanças, Logística, Produção, Marketing e Vendas*. São Paulo, SP: Livros Técnicos e Científicos, 2007.

Colorni, A.; Dorigo, M.; Maniezzo, V. & et al., Distributed optimization by ant colonies. In: Varela, F.J. & Bourgine, P. (Eds.), *Proceedings of the First European Conference on Artificial Life*. Cambridge, USA: MIT Press, p. 134–142, 1992.

Cormen, T.H.; Leiserson, C.E.; Rivest, R.L. & Stein, C., *Introduction to Algorithms*. 3a edição. Cambridge, USA: MIT Press, 2009.

Crawford, B. & Castro, C., Integrating lookahead and post processing procedures with aco for solving set partitioning and covering problems. In: Rutkowski, L.; Tadeusiewicz, R.; Zadeh, L. & Zurada, J. (Eds.), *Artificial Intelligence and Soft Computing*. v. 4029 de *Lecture Notes in Computer Science*, p. 1082–1090, 2006.

Croes, A., A method for solving traveling salesman problems. *Operations Research*, 6:791–812, 1958.

Deng, G.F. & Lin, W.T., Ant colony optimization-based algorithm for airline crew scheduling problem. *Expert Systems and Applications*, 38(5):5787–5793, 2011.

Desrochers, M. & Soumis, F., A column generation approach to the urban transit crew scheduling problem. *Transportation Science*, 23(1):1–13, 1989.

Dorigo, M., *Optimization, Learning and Natural Algorithms [em italiano]*. PhD thesis, Politecnico di Milano, Dipartimento di Elettronica ed Informatica, 1992.

Dorigo, M. & Caro, G.D., Ant colony optimization: A new meta-heuristic. In: *Proceedings of the Congress on Evolutionary Computation*. Piscataway, USA: IEEE Press, p. 1470–1477, 1999.

Dorigo, M. & Gambardella, L., Ant colonies for the traveling salesman problem. *BioSystems*, 43:73–81, 1997a.

Dorigo, M. & Gambardella, L.M., Ant colony system: a cooperative learning approach to the traveling salesman problem. *IEEE Transactions Evolutionary Computation*, 1(1):53–66, 1997b.

Dorigo, M.; Maniezzo, V. & Colorni, A., The ant system: Optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics - Part B*, 26(1):29–41, 1996.

Dorigo, M. & Stützle, T., *Ant Colony Optimization*. Cambridge, USA: MIT Press, 2004.

Eilon, S.; Watson-Gandy, C.D.T. & Christofides, N., *Distribution management: mathematical modelling and practical analysis*. London, UK: Griffin, 1971.

Fisher, M., The Lagrangian relaxation method for solving integer programming problems. *Management Science*, 27(1):1–18, 1981.

Flores, P.F.; Neto, H.C. & Marques-Silva, J.P., On applying set covering models to test set compaction. In: *Proceedings of Ninth Great Lakes Symposium on VLSI*. Washington, USA: IEEE Computer Society, p. 8–11, 1999.

Hadji, R.; Rahoual, M.; Talbi, E. & Bachelet, V., Ant colonies for the set covering problem. In: *Proceedings of ANTS2000 – From Ant Colonies to Artificial Ants: a Series of International Workshops on Ant Algorithms*. p. 63–66, 2000.

Housos, E. & Elmroth, T., Automatic optimization of subproblems in scheduling airline crews. *Interfaces*, 27(5):68–77, 1997.

Jacobs, L.W. & Brusco, M.J., A local-search heuristic for large set-covering problems. *Naval Research Logistics*, 42(7):1129–1140, 1995.

Leguizamon, G. & Michalewicz, Z., A new version of ant system for subset problems. In: *Proceedings of the Congress on Evolutionary Computation*. Piscataway, USA: IEEE Press, v. 2, 1999.

Lessing, L.; Dumitrescu, I. & Stützle, T., A comparison between ACO algorithms for the set covering problem. In: Dorigo, M.; Birattari, M.; Blum, C.; Gambardella, L.M.; Mondala, F. & Stützle, T. (Eds.), *Ant Colony Optimizaton and Swarm Intelligence*. Heidelberg, Germany: Springer-Verlag, v. 3172 de *Lecture Notes in Computer Science*, p. 1–12, 2004.

Lin, S., Computer solutions of the traveling salesman problem. *Bell System Technical Journal*, 44(10):2245–2269, 1965.

Lin, S. & Kernighan, B.W., An effective heuristic algorithm for the travelling-salesman problem. *Operations Research*, 21:498–516, 1973.

Maniezzo, V., Exact and approximate nondeterministic tree-search procedures for the quadratic assignment problem. *INFORMS Journal on Computing*, 11(4):358–369, 1999.

Maniezzo, V. & Colorni, A., The ant system applied to the quadratic assignment problem. *IEEE Transactions on Knowledge and Data Engineering*, 11(5):769–778, 1999.

Mulati, M.H. & Constantino, A.A., Ant-line: A line-oriented aco algorithm for the set covering problem. In: *Proceedings of XXX International Conference of the Chilean Computer Science Society*. Curico, Chile, 2011.

de Oliveira, N.V., *Problema de Cobertura de Conjuntos - Uma Comparação Numérica de Algoritmos Heurísticos*. Dissertação de mestrado, Universidade Federal de Santa Catarina, Programa de Pós-Graduação em Engenharia de Produção, 1999.

Oliver, I.M.; Smith, D.J. & Holland, J.R.C., A study of permutation crossover operators on the traveling salesman problem. In: *Proceedings of the Second International Conference on Genetic Algorithms and their Applications*. Hillsdale, USA: L. Erlbaum Associates Inc., p. 224–230, 1987.

Rahoual, M.; Hadji, R. & Bachelet, V., Parallel ant system for the set covering problem. In: Dorigo, M.; Di Caro, G. & Sampels, M. (Eds.), *Ant Algorithms*. Heidelberg, Germany: Springer-Verlag, v. 2463 de *Lecture Notes in Computer Science*, p. 249–297, 2002.

Ramalhinho-Lourenço, H. & Serra, D., *Adaptive Approach Heuristics for the Generalized Assignment Problem*. Economics Working Papers 288, Department of Economics and Business, Universitat Pompeu Fabra, 1998.

Reinelt, G., *The Traveling Salesman: Computational Solutions for TSP Applications*. Berlin, Germany: Springer-Verlag, 1994.

Ren, Z.; Feng, Z.; Ke, L. & Chang, H., A fast and efficient ant colony optimization approach for the set covering problem. In: *Proceedings of the IEEE World Congress on Computational Intelligence*. Piscataway, USA: IEEE Press, p. 1839–1844, 2008.

Ren, Z.G.; Feng, Z.R.; Ke, L.J. & Zhang, Z.J., New ideas for applying ant colony optimization to the set covering problem. *Computers & Industrial Engineering*, 58(4):774–784, 2010.

Resende, M.G.C.; Thomas, & Feo, T.A., A GRASP for satisfiability. In: *Cliques, Coloring, and Satisfiability: The Second DIMACS Implementation Challenge*. American Mathematical Society, v. 26 de *DIMACS Series on Discrete Mathematics and Theoretical Computer Science*, p. 499–520, 1996.

Solnon, C. & Fenet, S., A study of ACO capabilities for solving the maximum clique problem. *Journal of Heuristics*, 12(3):155–180, 2006.

Stützle, T. & Hoos, H., Max-Min ant system and local search for combinatorial optimization. In: *Proceedings of 2nd International Conference on Metaheuristics*. Sophie-Antipolis, France, p. 1–15, 1997.

Stützle, T. & Hoos, H., Improvements on the ant system: Introducing the Max-Min ant system. In: Albrecht, R.; Smith, G. & Steele, N. (Eds.), *Proceedings of 3rd International Conference on Artificial Neural Networks and Genetic Algorithms*. Heidelberg, Germany: Springer-Verlag, p. 245–249, 1998.

Toregas, C.; Swain, R.; ReVelle, C. & Bergman, L., The location of emergency service facilities. *Operations Research*, 19:1363–1373, 1971.

Wade, A. & Salhi, S., An ant system algorithm for the mixed vehicle routing problem with backhauls. In: Resende, M.G.C.; de Sousa, J.P. & Viana, A. (Eds.), *Metaheuristics*. Norwell, USA: Kluwer Academic, p. 699–719, 2004.

Yagiura, M.; Kishida, M. & Ibaraki, T., A 3-flip neighborhood local search for the set covering problem. *European Journal of Operational Research*, 172(2):472–499, 2006.

Notas Biográficas

Mauro Henrique Mulati é graduado em Informática (Universidade Estadual de Maringá – UEM, 2005) e mestre em Ciência da Computação (UEM, 2009). Atualmente é Professor Assistente do Departamento de Ciência da Computação da Universidade Estadual do Centro-Oeste – UNICENTRO.

Ademir Aparecido Constantino é graduado em Matemática (UEM, 1990), mestre e doutor em Engenharia de Produção na área de Otimização e Simulação (UFSC, 1993 e 1997, respectivamente), além de Pós-doutorado na Universidade de Nottingham, Inglaterra. Atualmente é professor titular do Departamento de Informática da Universidade Estadual de Maringá.

Anderson Faustino da Silva é graduado em Ciência da Computação (Universidade Estadual do Oeste do Paraná – UNIOESTE, 1999), mestre e doutor em Engenharia de Sistemas e Computação (COPPE/UFRJ, 2003 e 2006, respectivamente). Atualmente é Professor Adjunto do Departamento de Informática da Universidade Estadual de Maringá.