Universidade do Minho
Escola de Engenharia
Departamento de Informática

Otimização meta-heurística: Colónia de formigas 2017/2018

Paulo Novais, Tiago Pinto

ISLab
Synthetic Intelligence Lab

Otimização

- **Optimização**
  - Estudo de problemas em que se visa exprimir em termos matemáticos, o desejo de resolver um problema da melhor maneira
  - Estudo de problemas em que se procura minimizar ou maximizar uma função através da escolha sistemática dos valores de variáveis reais ou inteiras dentro de um conjunto viável

- **Investigação Operacional**
  - É um ramo interdisciplinar da matemática aplicada que faz uso de modelos matemáticos, estatísticos e de algoritmos na ajuda à tomada de decisões
  - É usada para analisar/modelar sistemas complexos do mundo real, com o objetivo de melhorar ou otimizar o seu desempenho

ISLab
Synthetic Intelligence Lab

Programação Linear e Quadrática

- **Utilizada quando a função objetivo e restrições são funções lineares ou quadráticas**
- **Métodos mais usados são o Simplex e o Método dos Pontos Interiores**
  - **Desvantagens:**
    - Dificilmente os problemas reais podem ser representados por funções lineares e variáveis inteiras
    - A utilização destes métodos requer a simplificação dos problemas
    - Incapazes de lidar com problemas de grande dimensão
  - **Vantagens:**
    - Garantir a solução ótima
    - Rápidos para problemas de pequena dimensão
    - Podem ser usados facilmente com outros métodos
    - Grande flexibilidade em termos de desenvolvimento dos modelos

ISLab
Synthetic Intelligence Lab

Programação Não Linear

- **Muito Semelhante à Programação Linear mas neste caso a função objetivo e restrições podem ser Não Lineares (NL)**
- **Existem vários métodos para resolver problemas NL (Quasi-Newton e o Método dos Pontos Interiores modificado)**
  - **Desvantagens:**
    - Frequentemente os problemas exigem variáveis inteiras para a sua representação
    - A aplicação dos métodos é muito exigente em termos computacionais pelo que é muito difícil resolver problemas reais, nomeadamente pela sua dimensão
  - **Vantagens:**
    - Permitem garantir boas soluções para problemas Não-Lineares

ISLab
Synthetic Intelligence Lab

Programação Inteira e Inteira Mista

- **No caso da Programação Inteira (PI) são admitidas unicamente variáveis inteiras, e no caso da Programação Inteira Mista (PIM) são admitidas variáveis inteiras e não inteiras**
- **Os métodos mais usados são o Branch and Bound e os Planos de Corte**
  - **Desvantagens:**
    - Requisitos elevados em termos de recursos computacionais (memória e capacidade de processamento)
    - Dificuldades de resolução de problemas reais
    - Quando comparados com alguns tipos de métodos são normalmente mais lentos
  - **Vantagens:**
    - Admitem variáveis Inteiras necessárias para representar estados e decisões (ex. Ligado, Desligado, comprar, não comprar, etc.)
    - Os modelos são legíveis não sendo necessário um grande nível de abstração para a sua compreensão
    - Existem muitas ferramentas comerciais para o desenvolvimento de aplicações

ISLab
Synthetic Intelligence Lab

Programação Dinâmica

- **Um dos primeiros métodos de otimização oriundos da Programação Matemática. Baseia-se na divisão do problema em sub-problemas para os quais tenta encontrar a solução ótima**
  - **Vantagens:**
    - Garante a solução ótima
    - Para algumas classes de problemas, para as quais a PD é adequada, pode ser muito eficiente, encontrando melhores soluções que os demais
  - **Desvantagens:**
    - Dado que se baseia num tipo de exploração exaustiva do espaço de pesquisa pode tornar-se impraticável o seu uso em problemas de grande dimensão.
    - Exige algum esforço para o desenvolvimento dos modelos e aplicação do método.

ISLab
Synthetic Intelligence Lab

Desafios

- **Na otimização de problemas complexos muitas vezes não é possível a aplicação de métodos matemáticos**
  - Impossibilidade de representação de problemas reais
  - Grande número de variáveis, restrições e/ou complexidade intrínseca do problema levam os métodos clássicos a necessitar de tempos demasiado elevados para identificar a solução ótima

- **Necessidade de soluções que garantam um balanço adequado entre a Eficácia (qualidade da solução) e Eficiência (tempo de execução / utilização de recursos computacionais) na resolução dos problemas**

ISLab
Synthetic Intelligence Lab

Vamos jogar um jogo...

- Identificar o máximo de uma função [0, 100]
- 10 tentativas

ISLab
Synthetic Intelligence Lab

Exploration vs Exploitation

- **A resolução de problemas complexos requer a obtenção das melhores soluções possíveis em tempo útil**
- **Para isso, ao invés da experimentação de todas as soluções possíveis (garantindo a solução ótima), é necessária a identificação de soluções o mais próximas quanto possível da solução ótima, num número limitado de tentativas**
- **É então, essencial um balanceamento adequado entre:**
  - ***Exploration*:** exploração geral do espaço de procura
  - ***Exploitation*:** pesquisa focada nas zonas mais promissoras

ISLab
Synthetic Intelligence Lab

Exploration vs Exploitation

- ***Exploration* sem *Exploitation* permite ter uma visão geral do espaço de procura, mas sem chegar muito próximo do valor ótimo**
- **Partir para a *Exploitation* de uma zona numa fase inicial do processo de procura pode levar a que a procura fique presa num ótimo local**
- **Este balanceamento é normalmente gerido com sucesso através de métodos Meta-heurísticos**

ISLab
Synthetic Intelligence Lab

Meta-heurísticas

- **Uma meta-heurística é um método heurístico para resolver de forma genérica problemas de otimização**
- **Meta-heurísticas são geralmente aplicadas a problemas para os quais não se conhece algoritmos eficientes**
- **Utilizam combinação de escolhas aleatórias e conhecimento histórico dos resultados anteriores adquiridos pelo método para se guiarem e realizar suas buscas pelo espaço de pesquisa em vizinhanças dentro do espaço de procura, o que evita ótimos locais**

Meta-heurísticas

- **Normalmente inspiradas em fenómenos da natureza**
- **Pela sua componente aleatória, são não-determinísticos**
- **Não garantem a identificação da solução ótima**
  - mas sim uma solução próxima
  - num tempo de execução rápido
  - utilizando menos recursos computacionais que as técnicas tradicionais

Meta-heurísticas

- **Pesquisa local vs pesquisa global**
  - Algumas meta-heurísticas aplicam métodos de pesquisa local, onde as novas soluções exploradas são "vizinhas" de soluções anteriores (e.g. *Simulated Annealing, Tabu Search*)
  - Outras meta-heurísticas distribuem o processo de procura por todo o espaço de procura (normalmente através de abordagens baseadas em populações)

- **Solução única vs *Population-based***
  - As abordagens de solução única, são iterativas, e orientam o processo de procura através da melhoria da solução anterior
  - As abordagens baseadas em populações utilizam uma pesquisa em paralelo por parte de vários membros da população, podendo, ou não, existir a troca de informação entre os indivíduos (e.g. *particle swarm optimization, genetic algorithms, ant colony optmization*)

- **Ao longo dos tempos foram sendo propostas diferentes métodos meta-heurísticos, sendo que atualmente existem centenas de métodos alternativos, embora partilhando das características fundamentais. Alguns dos mais relevantes são:**
  - Particle swarm optimization
  - Genetic algorithms
  - Simulated annealing
  - Tabu search
  - Artificial imune systems
  - Ant colony optimization

Ant Colony Optimization (ACO)

- **A meta-heurística *Ant Colony Optimization* (ACO) baseia-se no comportamento real das formigas**
  - Comportamento permite encontrar o menor caminho entre uma fonte de comida e a respetiva colónia
  - Este fenómeno ocorre porque, durante a sua trajetória, as formigas depositam no caminho uma substancia chamada feromona. Ao optarem por uma trajetória, escolhem aquela que possui a maior quantidade de feromona, pois é a trajetória que o maior número de formigas já realizou
    - Sugere que seja a melhor trajetória, ou por ser mais curta ou por ser a trajetória mais segura (e.g. que evita predadores)

ISLab
Synthetic Intelligence Lab

Ant Colony Optimization (ACO)

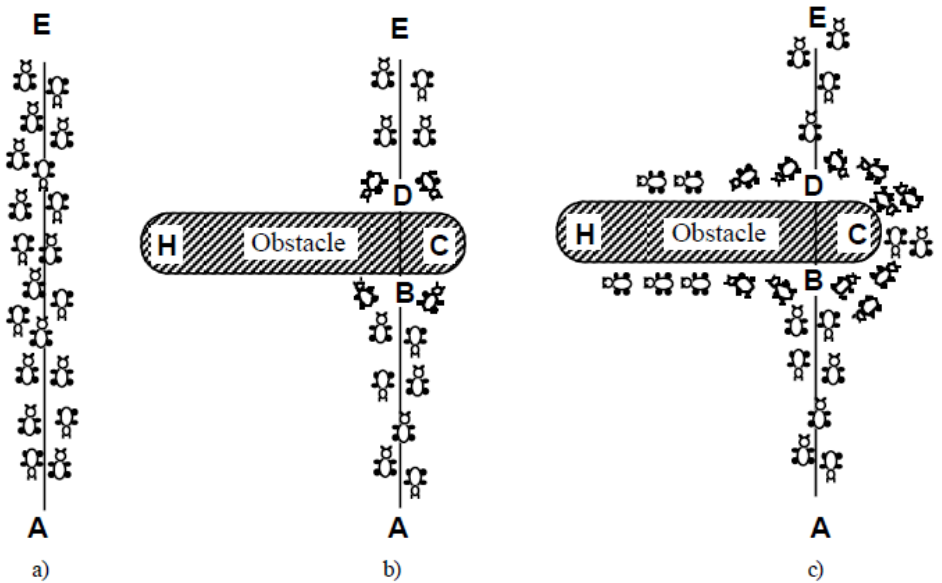


Fig. 1. An example with real ants.

a) Ants follow a path between points A and E.
b) An obstacle is interposed; ants can choose to go around it following one of the two different paths with equal probability.
c) On the shorter path more pheromone is laid down.

M. Dorigo, V. Maniezzo and A. Colorni, "Ant system: optimization by a colony of cooperating agents," in *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 26, no. 1, pp. 29-41, Feb 1996.

Ant Colony Optimization (ACO)

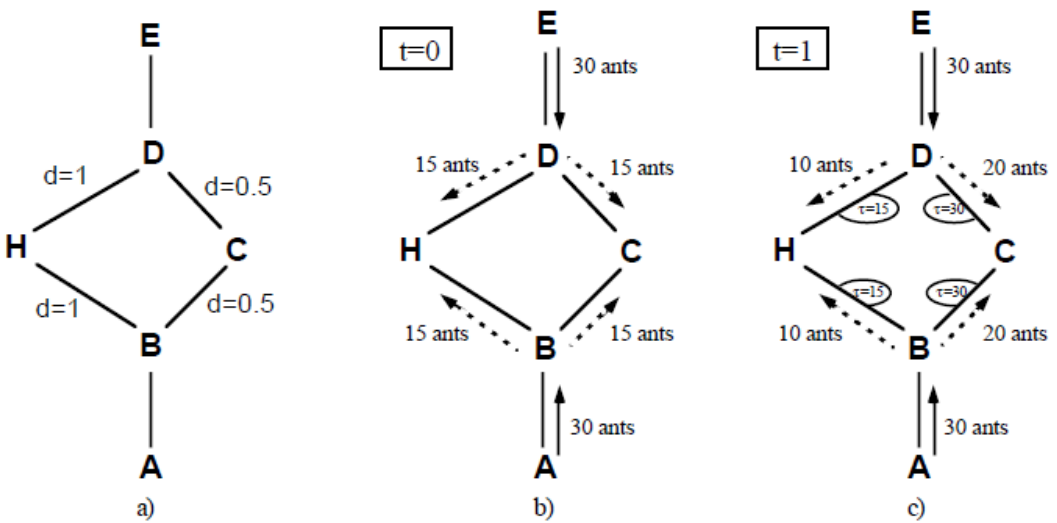


Fig. 2. An example with artificial ants.

a) The initial graph with distances.
b) At time t=0 there is no trail on the graph edges; therefore, ants choose whether to turn right or left with equal probability.
c) At time t=1 trail is stronger on shorter edges, which are therefore, in the average, preferred by ants.

M. Dorigo, V. Maniezzo and A. Colorni, "Ant system: optimization by a colony of cooperating agents," in *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 26, no. 1, pp. 29-41, Feb 1996.

ISLab
Synthetic Intelligence Lab

- **A ideia fundamental é então que, se num certo momento, uma formiga tem de escolher entre diferentes caminhos, aqueles que foram mais vezes escolhidos por formigas anteriores são escolhidos com maior probabilidade.**
- **Assim, caminhos com grande afluência de formigas são os caminhos mais curtos**

Ant Colony Optimization (ACO)

- **O ACO tem vindo a ser aplicado em diversos domínios, incluindo:**
  - Problemas de escalonamento
  - Problemas de mobilidade de veículos
  - Processamento de imagem
  - Classificação
  - entre muitos outros

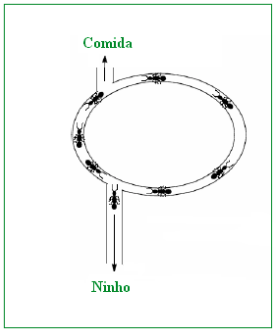
- **No entanto, os domínios de aplicação onde o ACO se tem mostrado mais robusto são:**
  - Como heurística para geração de solução inicial para outros métodos mais complexos
  - Na resolução de problemas relacionados com grafos e distâncias, e.g. problema do Caixeiro-Viajante

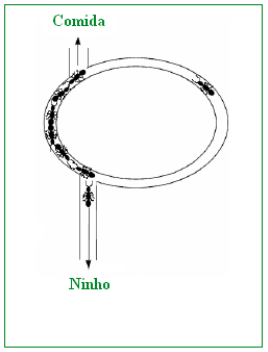
Caixeiro-Viajante

- **Problema do Caixeiro-Viajante (Traveling Salesman Problem – TSP)**
  - conjunto de n cidades
  - matriz n x n de distâncias
  - objetivo – percorrer todas as cidades uma única vez e voltar à cidade de partida, minimizando a distância total percorrida
  - **Quantas soluções admissíveis? (n-1)!**
    - 5 cidades ⇒ 4! = 24 percursos possíveis
    - 10 cidades ⇒ 9! = 362 880 percursos possíveis
    - 25 cidades ⇒ 24! = 6.2 x 1023 percursos possíveis

ISLab
Synthetic Intelligence Lab

Algoritmo ACO





- **No início as formigas são deixadas livres para escolher o caminho. Não há ainda feromona**
- **As formigas convergem para um dos caminhos com igual probabilidade**
- **Devido a flutuações aleatórias, um dos caminhos terá mais feromona e atrairá as formigas com maior probabilidade**
- **Usando caminhos de tamanhos diferentes, as formigas convergem para o mais curto**
  - O caminho curto é percorrido em menos tempo, fazendo com que mais formigas o consigam percorrer no mesmo tempo. Logo, mais é depositada mais feromona
  - As formigam escolhem, com maior probabilidade, o caminho mais curto (com mais feromona).
  - **Formigas que usam o menor caminho vão e voltam mais rapidamente**
    - **Ocorre maior depósito de feromonas no menor caminho**
    - **No final as formigas usam sempre o menor caminho**
    - **O caminho mais curto de todos os visitados acaba por ser o que é percorrido em menor tempo, logo acaba por receber maior quantidade de feromona, atraindo mais e mais formigas**

Algoritmo ACO

- **O algoritmo de otimização por colónias de formigas "imita" o comportamento de uma colónia de formigas através do "lançamento" sucessivo de um determinado número de sub-colónias (NS) cada uma com um determinado número de formigas (NF)**
- **Cada formiga tem que construir uma solução admissível para o problema em causa. Quando todas as formigas da sub-colónia tiverem completado a sua solução, os trilhos de feromona são atualizados**

Repetir (de s=1 até s=N_S)

Repetir (de f=1 até f=N_F)

Formiga f constrói uma solução

Actualização dos trilhos de feromona: $\tau(i,k) \leftarrow (1-\rho)\ \tau(i,k) + \sum \Delta\tau_f(i,k)$

Nível de feromona entre os nós i e k

Evaporação de alguma quantidade de feromona

Deposição de feromona entre os nós i e k pelas formigas que utilizaram o arco (i,k)

Algoritmo ACO

- **Em cada iteração, as formigas decidem, de acordo com uma medida probabilística, qual o caminho a seguir, de entre os não visitados**
- **Cada formiga tem uma memória, também chamada de lista tabu, que armazena os caminhos percorridos e evita que um caminho seja visitado pela mesma formiga mais de uma vez**
- **A probabilidade de escolha de um caminho é proporcional ao rasto de feromona e à sua atratividade - que varia de acordo com o tipo e a modelação do problema em que o algoritmo está a ser aplicado**
- **Se a formiga já percorreu aquele caminho, a probabilidade de escolha é zero; caso contrário, é positiva.**

ISLab
Synthetic Intelligence Lab

Algoritmo ACO – escolha do caminho

- **A escolha do caminho a ser seguido pela formiga (construção da solução) é feita segundo a Probabilidade de Transição entre caminhos:**

$$p_k(r,s) = \begin{cases} \dfrac{\left[\tau_{(r,s)}\right]^{\alpha} \cdot \left[\eta_{(r,s)}\right]^{\beta}}{\sum_{u \notin M_k} \left[\tau_{(r,u)}\right]^{\alpha} \cdot \left[\eta_{(r,u)}\right]^{\beta}}, & \text{se } s \notin M_k \\ 0, & \text{caso contrário} \end{cases}$$

Quantidade de feromona

Atratividade do caminho (normalmente 1/distância)

Parâmetro para controlar a influência da atratividade

Parâmetro para controlar a influência da feromona

Algoritmo ACO – feromonas

- **Caso um caminho não tenha sido percorrido por formigas, a deposição de feromona no mesmo é zero; caso contrário, é positiva**
- **É dito que uma formiga mudou de "estado" quando se move de uma solução para outra**
- **No final de cada iteração, uma taxa de evaporação remove parte da feromona, reduzindo a quantidade da substância nos caminhos. Isso evita que as formigas fiquem presas em ótimos locais e, ao mesmo tempo, diminui a probabilidade de escolha de caminhos que não foram utilizados recentemente**
- **A deposição do feromona é feita no final de cada iteração**
  - Considera-se que, no final de cada iteração, as formigas fazem o caminho inverso da geração da solução, mas, desta vez, depositando o feromona. A quantidade de feromona associada a cada caminho representa a aprendizagem da colónia no decorrer do algoritmo.

ISLab
Synthetic Intelligence Lab

Algoritmo ACO – feromonas

- **Na atualização das feromonas ocorrem dois eventos:**
  - A evaporação
    - Evita que a feromona acumulada cresça indefinidamente
    - Permite esquecer más decisões do passado da pesquisa
  - O depósito de feromonas de todas as formigas

$$\tau_{xy} \leftarrow (1-\rho)\tau_{xy} + \sum_k \Delta\tau_{xy}^k$$

Quantidade de feromona

Coeficiente de evaporação da feromona

Quantidade de feromona depositada pela formiga *k*

Constante

Distância

$$\Delta\tau_{xy}^k = \begin{cases} Q/L_k & \text{if ant } k \text{ uses curve } xy \text{ in its tour} \\ 0 & \text{otherwise} \end{cases}$$

ISLab
Synthetic Intelligence Lab

Referências

- Dorigo, M., Stutzle, T., (2003), The Ant Colony Optimization Metaheuristic: Algorithms, Applications, and Advances. In Glover, F., Kochenberger, G. Handbook on Metaheuristics, Kluwer, Cap. 9.
- Dorigo M., Di Caro, G., Cambardella L. M., (1999), Ant Algorithms for Discrete Optimization. Artificial Life, 5(2):137-172. Also available as Technical Report No. 98-10 (IRIDIA), Université Libre de Bruxelles, Belgium.
- Dorigo, M., Maniezzo V. and Colorni, A."Ant system: optimization by a colony of cooperating agents," in IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), vol. 26, no. 1, pp. 29-41, Feb 1996.
- Página sobre Ant Colony Optimization, desenvolvida por MarcoDorigo: http://iridia.ulb.ac.be/~mdorigo/ACO/index.html

Universidade do Minho
Escola de Engenharia
Departamento de Informática

Otimização meta-heurística: Colónia de formigas 2017/2018

Paulo Novais, Tiago Pinto