Ant-System: Propostas de Melhoramento

Prof. Ademir A. Constantino
Departamento de Informática
Universidade Estadual de Maringá
www.din.uem.br/~ademir

O Algoritmo para o PCV (revisão)

- Considere um conjunto de n cidades, o objetivo PCV é encontrar o menor caminho (*tour*) passando por cada cidade pelo menos uma vez e retornando à cidade de origem.

- Seja $b_i(t)$ (i=1, 2, ..., n) o número de formigas na cidade i no tempo t, portanto, $m = \sum_{i=1}^{n} b_i(t)$, é o número total de formigas.

- Depois de uma formiga ter encontrado um *tour*, uma substância denominada de **resíduo** (feromônio) é depositada nas arestas visitadas. A quantidade de resíduo na aresta (i, j) no tempo t é representada por $\tau_{ij}(t)$.

O Algoritmo para o PCV

- Cada *iteração* é definida por $(t+n)$. A cada iteração a intensidade de resíduo na aresta é atualizado por:

$$\tau_{ij}(t+n) = \rho\tau_{ij}(t) + \Delta\tau_{ij} \qquad (1)$$

sendo

ρ um coeficiente de persistência tal que $(1-\rho)$ representa a taxa de evaporação do resíduo entre os tempos t e $t+n$.

$$\Delta\tau_{ij} = \sum_{k=1}^{m} \Delta\tau_{ij}^{k} \qquad (2)$$

é a quantidade da substância a ser depositada na aresta (i,j) pela k-ésima formiga .

O Algoritmo para o PCV

- Quantidade de resíduo depositado na aresta (modelo "*ant-cycle*").

$$\Delta \tau_{ij}^{k} = \begin{cases} Q / L_k \text{ se a k - ésima formiga usou a aresta } (i, j) \\ \qquad \text{em seu tour (entre o tempo } t \text{ e } t + n). \\ 0 \text{ caso contrário} \end{cases} \qquad (3)$$

onde Q é uma constante e L_k é o comprimento do tour da k-ésima formiga.

O Algoritmo para o PCV

- Os vértices visitados pela *k*-ésima formiga são colocados em uma lista **tabu**$_k$. Define-se por *visibilidade* $\eta_{ij} = 1/d_{ij}$. Assim, a probabilidade de transição de uma cidade *i* para a cidade *j* pela *k*-ésima formiga é definida como:

$$p_{ij}^{k}(t) = \begin{cases} \dfrac{[\tau_{ij}(t)]^{\alpha} \cdot [\eta_{ij}]^{\beta}}{\sum_{l \in allowed_k} [\tau_{il}(t)]^{\alpha} \cdot [\eta_{il}]^{\beta}} & \text{se } j \in allowed_k \\ 0 & \text{caso contrário} \end{cases} \qquad (4)$$

sendo $allowed_k$ o conjunto de cidades que ainda não foram visitadas pela *k*-ésima formiga e $\alpha\beta$ significam a importância relativa do resíduo e a atratividade.

Algoritmo: Resumo

- O algoritmo é construtivo
- Inicialização: t=0 e cada formiga é posicionada em uma cidade diferente.
- Em todo tempo t cada formiga escolhe probabilisticamente a próxima cidade que ela estará no tempo t+1;
- Uma *iteração* do algoritmos corresponde a m movimentos realizados pelas m formigas no intervalo (t, t+1);
- Em n iterações (denominado de *ciclo*) cada formiga terá completado um *tour*.

Melhoramentos sobre o *Ant System*

Stützle, Thomas and Dorigo, M. Maniezzo. **ACO Algorithms for Traveling Salesman Problem.** In: *Evolutionary Algorithms in Engineering and Computer Science: Recent Advances in Genetic Aglorithms, Evolutions Strategies. Evolutionary Programing, Genetic Programming and Industrial Applications.* Johs Wiley & Sons, 1999

Este artigo apresenta alguns melhoramentos (*improvement*) sobre o *Ant System* original.

Melhoramentos sobre o *Ant System*

- Nesse artigo o paradigma *Ant System* passou a ser chamado de ACO (*Ant Colony Optimization*).
- Apresenta a proposta de algoritmo ACO-híbrido:

```
procedure ACO algorithm for TSPs
    Set parameters, initialize pheromone trails
    while (termination condition not met) do
        ConstructSolutions
        ApplyLocalSearch          % optional
        UpdateTrails
    end
end ACO algorithm for TSPs
```

Melhoramentos sobre o *Ant System*

- ACS (*Ant Colony System*) para o PCV.
  * Faz uma conexão do ACO com Ant-Q (Q-learning – aprendizagem por reforço)
- Diferenças entre o ACS e o AS original:
  * O ACS utiliza uma regra de escolha gulosa mais "agressiva";
  * O feromônio é atualizado somente nos arcos da melhor solução global encontrada;
  * Cada vez que uma formiga utiliza uma arco (i, j) para se locomover, ela retira uma quantidade de feromônio desse arco.

Detalhes do Algoritmo ACS

- Regra de Transição:
  A formiga K estando no vértice i, então ela move-se para o vértice j com base no maior valor de : $\tau_{ij}(t)[n_{ij}]^{\beta}$
- Atualização Global do feromônio:
  O feromônio é depositado sobre as arestas que participam da melhor solução entre todas as formigas com base na seguinte expressão: $\tau_{ij}(t+n) = (1-\rho)\tau_{ij}(t) + \rho\Delta\tau_{ij}^{gb}$
  Onde

$$\Delta\tau_{ij}^{gb} = 1/L^{gb};$$

L^{gb} = custo da melhor solução;

ρ = taxa de evaporação.

Detalhes do Algoritmo ACS

- Atualização Local do feromônio:
  Esta atualização ocorre imediatamente após uma formiga ter escolhido um arco (i, j):

$$\tau_{ij} = (1-\xi)\tau_{ij} + \xi\tau_0$$

Onde

$\xi, 0 < \xi < 1$ e

τ_0 são dois parâmetros.

Melhoramentos sobre o *Ant System*

- MMAS (Max-Min *Ant System*) para o PCV.
- Diferenças entre o MMAS e o AS original:
  * O ACS utiliza uma regra de escolha semelhante ao AS original.
  * O feromônio é atualizado somente nos arcos da melhor solução encontrada;
  * É definido um limite mínimo e máximo para o feromônio
  * Na inicialização, o feromônio recebe o limite máximo de feromônio permitido, causando uma exploração mais intensa no início do algoritmo.

Detalhes do Algoritmo MMAS

- Atualização do feromônio:
  Esta atualização é feita de maneira um pouco parecida como ACS:
  $$\tau_{ij}(t+n) = (1-\rho)\tau_{ij}(t) + \Delta\tau_{ij}^{best}$$
- Limites do feromônio
  O feromônio é limitado por $\tau_{\min} \leq \tau_{ij} \leq \tau_{\max}$
  O objetivo é evitar a estagnação da busca.

Melhoramentos sobre o *Ant System*

AS_{rank} (Ant System baseado em *ranking*)

(Inspirado na idéia elitista de Algoritmos Genéticos)

- Ordenar um subconjunto das melhores formigas baseado no custo da solução alcançada.
- A quantidade de feromônio que uma formiga pode depositar é ponderada pela sua posição no *rank*.

Adaptar a ACO para outros problemas.

Como criar um algoritmo ACO para o meu problema de otimização?

- Inicie pesquisando por algoritmos construtivos clássicos, por exemplo, os algoritmo gulosos.
- Utilize a função gulosa para definir a *visibilidade*. Funções gulosas para o problema encontradas na literatura podem ser boas fontes de partida para definir a visibilidade.
- Defina a probabilidade de transição:
  * a probabilidade pode relacionada com todos elementos já presentes na solução em construção, ou apenas com o último elemento inserido, ou com ninguem.
  * Observe que a probabilidade de transição, na forma clássica do AS para o PCV, está relacionada somente com o último vértice adicionado à solução.