Swarm Intelligence
Travelling Salesman Problem and Ant System

Christian Camacho Villalón
christian.camacho.villalon@ulb.ac.be
Federico Pagnozzi
federico.pagnozzi@ulb.ac.be

IRIDIA – Université Libre de Bruxelles (ULB)
Bruxelles, Belgium
Outline

1. Concepts review
2. Travelling salesman problem
   • Problem definition
   • Examples
3. Ant System Algorithm
   • Description
   • Application to TSP
4. Class exercise
5. Practical exercise
Concept review

• Optimization problems
• Objective function
• Search space
  – Local / global optima
• Searching
  – Exact vs. approximation methods
  – Constructive vs. perturbative
• Exploration and exploitation
Travelling Salesman Problem
Informal definition

• Given a set of cities, a salesman needs to find a shortest tour that takes him through all cities just once and then back home.
Main reasons for choosing the TSP:

- It is a classical **combinatorial optimization problem**.
- It is **NP hard**.
- It is the problem to which the Ant System algorithm was first applied.
- Popular test bed for new algorithms.
The TSP can be modelled as a Graph $G(N,A)$ where:

- $N$ is the set of nodes representing the cities
- $A$ is the set of edges
- Each edge is assigned a cost value (length) $d$
  - $d_{ij}$ is the edge cost, or the length from city $i$ to city $j$
Travelling Salesman Problem
Formal definition

Find a minimum length $f(\pi)$ Hamiltonian tour in a graph $G(N,A)$, where $n$ is number of nodes and $\pi$ is a permutation of the nodes indices.

$$f(\pi) = \sum_{i=1}^{n-1} d_{\pi(i)\pi(i+1)} + d_{\pi(n)\pi(1)}$$
The nearest neighbourhood heuristic is a simple greedy-type construction heuristic

- It starts from a randomly chosen city
- Greedy rule: select the closest city that is not yet visited

- Initial city: C  cost: 8
- Closest city: A  cost: 7
- Closest city: B  cost: 13
- Closest city: D  cost: 7
- Return city  cost: 9

Total: 44
Travelling Tournament Problem
First attempt to solve

- The nearest neighbour algorithm is easy to implement and executes quickly.
- Usually the last a few edges added are extremely large, due to the “greedy” nature.
- In some cases it even constructs the unique worst possible tour.
- How to generate a tour more intelligently?
  - Learn from the previous constructions!
Ant System

- **Ant System** is a basic ant-based algorithm.
- Ants visit the cities sequentially till they obtain a tour.
- Transition from city $i$ to $j$ depends on:
  - *Heuristic information* to visit city $j$ when in city $i$, associated to a static value based on the edge-cost (distance) $\eta_{ij}$
  - *Pheromone* that represents the learned desirability to visit city $i$ when in city $j$ associated to a dynamic value $\tau_{ij}$
Ant System
Stochastic Solution Construction

- Use **memory** to remember partial tours.
- Being at a city $i$ choose next city $j$ **probabilistically** among feasible neighbouring cities.
- Probabilistic choice depends on:
  - pheromone trails $\tau_{ij}$
  - heuristic information $\eta_{ij} = 1/d_{ij}$
- Random proportional rule at node $i$ is:

$$p_{ij}^k(t) = \frac{[\tau_{ij}(t)]^\alpha [\eta_{ij}]^\beta}{\sum_{l \in N_i^k} [\tau_{il}(t)]^\alpha [\eta_{il}]^\beta}, \text{ if } j \in N_i^k$$
Ant System

Pheromone Update

- Use **phermone evaporation** to avoid unlimited increase of pheromone trails and allow **forgetting** of earlier choices
  - Pheromone evaporation rate $0 < \rho \leq 1$

- Use **pheromone deposit** to positive feedback, reinforcing components of good solutions
  - Better solutions give more feedback
Ant System

Pheromone Update

• Example of pheromone update

$$\tau_{ij}(t) = (1-\rho) \cdot \tau(t-1) + \sum_{k=1}^{m} \Delta \tau_{ij}^k$$

$$\Delta \tau_{ij}^k = \frac{1}{L_k}, \text{ if arc}(i,j) \text{ is used by ant } k \text{ on its tour}$$

- $L_k$: Tour length of ant $k$
- $m$: number of ants
Ant System
Simple pseudo code

1. While !termination()
2. For k = 1 To m Do #m number of ants
3. \( \text{ants}[k][1] \leftarrow \text{SelectRandomCity}() \)
4. For i = 2 To n Do #n number of cities
5. \( \text{ants}[k][i] \leftarrow \text{ASDecisionRule}(\text{ants}, i) \)
6. EndFor
7. \( \text{ants}[k][n+1] \leftarrow \text{ants}[k][1] \) #to complete the tour
8. EndFor
9. UpdatePheromone(ants)
10. EndWhile
Ant System

Simple example

- For our example with \#ants=3, \( \alpha=2 \), \( \beta=1 \), \( \rho=0.5 \) and \( \tau_0=1 \)

- Heuristic Information

- Pheromone trails
Ant System
Simple example

- For ant #1 we start from city D (random), selection probabilities
  \[ p_{ij}^k(t) = \frac{[\tau_{ij}(t)]^\alpha \cdot [\eta_{ij}]^\beta}{\sum_{l \in N_i^k} [\tau_{il}(t)]^\alpha \cdot [\eta_{il}]^\beta} \]

- Select a city \( \rightarrow \) rand 0.80
  - City E selected

- Select a city \( \rightarrow \) rand 0.27
  - City B selected

- Select a city \( \rightarrow \) rand 0.88
  - City C selected

<table>
<thead>
<tr>
<th>( p_{ij} )</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>D</td>
<td>0.264</td>
<td>0.059</td>
<td>0.031</td>
<td>0.000</td>
<td>0.646</td>
</tr>
</tbody>
</table>

\[ [0, 0.264, 0.323, 0.354, 1] \]

<table>
<thead>
<tr>
<th>( p_{ij} )</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>E</td>
<td>0.267</td>
<td>0.227</td>
<td>0.506</td>
<td>0.000</td>
<td>0.000</td>
</tr>
</tbody>
</table>

\[ [0, 0.267, 0.494, 1] \]

<table>
<thead>
<tr>
<th>( p_{ij} )</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>0.843</td>
<td>0.000</td>
<td>0.157</td>
<td>0.000</td>
<td>0.000</td>
</tr>
</tbody>
</table>

\[ [0, 0.843, 1] \]
Ant System
Simple example

• First iteration we can have:
  – Ant #1: D-E-B-C-A-D
  – Ant #2: A-E-D-C-B-A
  – Ant #3: D-E-C-B-A-D

• Update the pheromone using this tours

$$
\tau_{ij}(t) = [1 - \rho] \cdot \tau(t - 1) + \sum_{k=1}^{m} \Delta \tau_{ij}^k
$$

• And then iterate

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>-</td>
<td>0.39</td>
<td>0.38</td>
<td>0.42</td>
<td>0.29</td>
</tr>
<tr>
<td>B</td>
<td>0.39</td>
<td>-</td>
<td>0.46</td>
<td>0.28</td>
<td>0.35</td>
</tr>
<tr>
<td>C</td>
<td>0.38</td>
<td>0.46</td>
<td>-</td>
<td>0.29</td>
<td>0.35</td>
</tr>
<tr>
<td>D</td>
<td>0.42</td>
<td>0.28</td>
<td>0.29</td>
<td>-</td>
<td>0.49</td>
</tr>
<tr>
<td>E</td>
<td>0.29</td>
<td>0.35</td>
<td>0.35</td>
<td>0.49</td>
<td>-</td>
</tr>
</tbody>
</table>
Ant System
Exercise #1

• Implement Ant System according to the provided template.
  - C++

• The following slides give a practical view of the Ant System algorithm procedures.
Ant System Algorithm

Solution Construction

Procedure ConstructSolutions ()
  For k = 1 To m Do
    For i = 1 To n Do
      ant[k].visited[i] ← false
    EndFor
  EndFor
  step ← 1
  For k = 1 To m Do
    r ← random{1, . . . , n}
    ant[k].tour [step] ← r
    ant[k].visited [r] ← true
  EndFor
  While (step < n) Do
    step ← step + 1
    For k = 1 To m Do
      ASDecisionRule(k, step)
    EndFor
  EndWhile
  For k = 1 To m Do
    ant[k].tour [n+1] ← ant[k].tour[1]
    ant[k].tour length ← ComputeTourLength(k)
  EndFor
EndProcedure
Ant System Algorithm

Decision Rule

1. **Procedure** ASDecisionRule(k, i)
   
   - *#k* ant identifier
   - *#i* counter for construction step
   - \( c \leftarrow \text{ant}[k].\text{tour}[i-1] \)
   - \( \text{sum\_prob} = 0.0 \)
   
   **For** \( j = 1 \) **To** \( n \) **Do**
   
   - **If** \( \text{ant}[k].\text{visited}[j] \) **Then**
     - \( \text{selection\_prob}[j] \leftarrow 0.0 \)
   
   - **Else**
     - \( \text{selection\_prob}[j] \leftarrow \text{choice\_info}[c][j] \)
     - \( \text{sum\_prob} \leftarrow \text{sum\_prob} + \text{selection\_prob}[j] \)
   
   **EndIf**
   
   **EndFor**
   
   - \( r \leftarrow \text{random}[0, \text{sum\_prob}] \)
   - \( j \leftarrow 1 \)
   
   **While** \( (p < r) \) **Do**
   
   - \( j \leftarrow j + 1 \)
   - \( p \leftarrow p + \text{selection\_prob}[j] \)
   
   **EndWhile**
   
   - \( \text{ant}[k].\text{tour}[i] \leftarrow j \)
   - \( \text{ant}[k].\text{visited}[j] \leftarrow \text{true} \)

2. **EndProcedure**
Ant System Algorithm

Pheromone Update

1  **Procedure** ASPheromoneUpdate()
2    Evaporate()
3    For  $k = 1$  To  $m$  Do
4      DepositPheromone($k$)
5    EndFor
6    ComputeChoiceInformation()
7  EndProcedure
Ant System Algorithm

Pheromone Update

1  Procedure Evaporate
2    For $i = 1$ To $n$ Do
3       For $j = i$ To $n$ Do
4          pheromone[$i$][$j$] $\leftarrow$ $(1-\rho) \cdot$ pheromone[$i$][$j$]
5          pheromone[$j$][$i$] $\leftarrow$ pheromone[$i$][$j$]
6          #pheromones are symmetric
7       EndFor
8    EndFor
9  EndProcedure
Ant System Algorithm

Pheromone Update

1 Procedure DepositPheromone(k)
2 # k ant identifier
3 $\Delta \tau \leftarrow 1/ant[k].\text{tour\_length}$
4 For $i = 1$ To $n$ Do
5 \hspace{1em} j $\leftarrow$ ant[k].tour[$i$]
6 \hspace{1em} l $\leftarrow$ ant[k].tour[$i+1$]
7 \hspace{1em} pheromone[$j$][$l$] $\leftarrow$ pheromone[$j$][$l$] + $\Delta \tau$
8 \hspace{1em} pheromone[$l$][$j$] $\leftarrow$ pheromone[$j$][$l$]
9 EndFor
10 EndProcedure
Ant System
Exercise #2

• Test and analyse the behaviour of the algorithm.
  – Modify some parameters:
    • Number of ants
    • $\alpha$, $\beta$, $\rho$

• What effect can you appreciate?
• What is the reason?