Swarm Intelligence
Travelling Salesman Problem and Ant System

Hayfa Hammami
haifa.hammami@ulb.ac.be
Christian Camacho Villalón
christian.camacho.villalon@ulb.ac.be

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

1. Concept 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 customer cities, a salesman from his home town needs to find a shortest tour that takes him through all customers just once and then back home.
Travelling Salesman Problem (TSP)

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.

● Often used to test new algorithms and variants.
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 $j$
Find a minimum length $f(\pi)$ Hamiltonian circuit of a graph $G(N,A)$, where $n$ is number of nodes and $\pi$ is a permutation of the nodes indices.

$$f(\pi) = \sum_{i=1}^{n-1} d_{\pi(i)\pi(i+1)} + d_{\pi(n)\pi(1)}$$
Travelling Tournament Problem
First attempt to solve – Constructive heuristic

- 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

![Graph showing the nearest neighbourhood heuristic process]

- Initial city: C
- Closest city: A cost: 8
- Closest city: B cost: 7
- Closest city: D cost: 13
- Closest city: E 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 behaviour 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:

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

• Use **pheromone 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 ants[k][1] ← SelectRandomCity()
4 For i = 2 To n Do #n number of cities
5 ants[k][i] ← ASDecisionRule(ants, i)
6 EndFor
7 ants[k][n+1] ← 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

\[
\begin{array}{c|ccccc}
\eta_{ij} & A & B & C & D & E \\
\hline
A & - & 1/1 & 1/2 & 1/2 & 1/6 \\
B & 1/1 & - & 1/6 & 1/8 & 1/10 \\
C & 1/2 & 1/6 & - & 1/12 & 1/4 \\
D & 1/2 & 1/8 & 1/12 & - & 1/1 \\
E & 1/6 & 1/10 & 1/4 & 1/1 & - \\
\end{array}
\]

\[
\begin{array}{c|ccccc}
\tau_{ij} & A & B & C & D & E \\
\hline
A & - & 0.56 & 0.66 & 0.60 & 0.50 \\
B & 0.56 & - & 0.60 & 0.56 & 0.60 \\
C & 0.66 & 0.60 & - & 0.50 & 0.56 \\
D & 0.60 & 0.56 & 0.50 & - & 0.66 \\
E & 0.50 & 0.60 & 0.56 & 0.66 & - \\
\end{array}
\]
Ant System
Simple example

- For ant #1 we start from city D (random), selection probabilities

\[ p_{ij}^k(t) = \frac{[\tau_{ij}(t)]^\alpha \cdot [\eta_{ij}]^\beta}{\sum_{l \in N_i} [\tau_{il}(t)]^\alpha \cdot [\eta_{il}]^\beta} \]

<table>
<thead>
<tr>
<th>pij</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 ]

- Select a city → rand 0.80
  - City E selected

<table>
<thead>
<tr>
<th>pij</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 ]

- Select a city → rand 0.27
  - City B selected

<table>
<thead>
<tr>
<th>pij</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 ]

- Select a city → rand 0.88
  - City C selected
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

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

<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>

- And then iterate
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

```plaintext
Procedure ConstructSolutions ()
    For k = 1 To m Do #m number of ants
        For i = 1 To n Do #n number of cities
            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

Procedure ASDecisionRule(k, i)
    #k ant identifier
    #i counter for construction step
    c ← ant[k].tour[i-1]
    sum_prob = 0.0
    For j = 1 To n Do
        If ant[k].visited[j] Then
            selection_prob[j] ← 0.0
        Else
            selection_prob[j] ← choice_info[c][j]
            sum_prob ← sum_prob + selection_prob[j]
        EndIf
    EndFor
    r ← random[0, sum_prob]
    j ← 1
    p ← selection_prob[j]
    While (p < r ) Do
        j ← j + 1
        p ← p + selection_prob[j]
    EndWhile
    ant[k].tour[i] ← j
    ant[k].visited[j] ← true
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] ← (1−ρ)·pheromone[i][j]
5         pheromone[j][i] ← pheromone[i][j]
#pheromones are symmetric
6     EndFor
7     EndFor
8 EndProcedure
Ant System Algorithm
Pheromone Update

1 Procedure DepositPheromone(k)
2     # k ant identifier
3     Δτ ← 1/ant[k].tour_length
4     For i = 1 To n Do
5         j ← ant[k].tour[i]
6         l ← ant[k].tour[i+1]
7         pheromone[j][l] ← pheromone[j][l] + Δτ
8         pheromone[l][j] ← 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?