Swarm Intelligence

Traveling Salesman Problem and Ant System

Leonardo Bezerra and Leslie Perez Caceres

IRIDIA – Université Libre de Bruxelles (ULB)
Bruxelles, Belgium
lperez@iridia.ulb.ac.be
leonardo@iridia.ulb.ac.be

Outline

1. Travelling salesman problem
   - Problem definition
   - Examples
2. Ant System Algorithm
   - Description
   - Applied to TSP
3. Practical exercise

Traveling 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.

Traveling Salesman Problem (TSP)

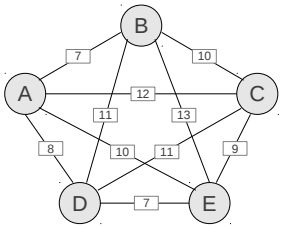
Main reasons for choosing the TSP:

- 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.

Traveling Salesman Problem
Formal Definition

The TSP can be modeled as a Graph ***G(N,A)*** where:

- ***N*** is the set of nodes representing the cities
- ***A*** is the set of arcs
- Each arc is assign a cost value (length) ***d***
  - d_{ij} is the arc cost, or the length from city *i* to city *j*



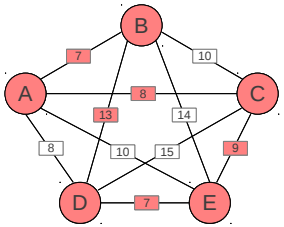
Traveling Salesman Problem
Formal definition

Find a minimum length ***f(π)*** *Hamiltonian circuit of a graph G(N,A), where n is number of nodes and* ***π*** *is a permutation of the nodes indices.*

$$f(\pi)=\sum_{i=1}^{n-1} d_{\pi(i)\pi(i+1)} + d_{\pi(n)\pi(1)}$$

Traveling Tournament Problem
First attempt to solve

- The **nearest neighborhood heuristic** is a simple greedy-type construction heuristic
  - It starts from a chosen city and always select the closest city that is not yet visited



- 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

- Lets see a more complex example

Traveling 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 desirability to visit city ***j*** when in city ***i***, associated to a static value based on the edge-cost (distance) η_{ij}
  - Pheromone that represents the learned desirability to visit city ***i*** when in city ***j*** associated to a dynamic value τ_{ij}

Ant System
Stochastic Solution Construction

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

$$p_{ij}^{k}(t) = [\tau_{ij}(t)]^{\alpha} \cdot [\eta_{ij}]^{\beta}$$

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 deposite to positive feedback, reinforcing components of good solutions
  - Better solutions give more feedback

Ant System
Pheromone Update

- Example of pheromone update

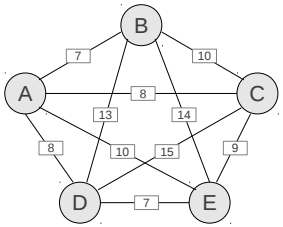
$$\tau_{ij}(t) = (1-\rho) \cdot \tau(t-1) + \sum_{k=1}^{m} \Delta\tau_{ij}^{k}$$

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

  - L_k**:** Tour length of ant *k*
  - ***m****: number of ants*

Ant System
Simple example

- For our example with #ants=3, α=1, β=5, ρ=0.5 and τ_0=1



  - Heuristic Information

nij	A	B	C	D	E
A	-	1/7	1/8	1/8	1/10
B	1/7	-	1/10	1/13	1/14
C	1/8	1/10	-	1/15	1/9
D	1/8	1/13	1/15	-	1/7
E	1/10	1/14	1/9	1/7	-

  - Pheromone trails

tij	A	B	C	D	E
A	-	1	1	1	1
B	1	-	1	1	1
C	1	1	-	1	1
D	1	1	1	-	1
E	1	1	1	1	-

Ant System
Simple example

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

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

pij	A	B	C	D	E
C	0.2498	0.2520	-	0.2637	0.2493

- Select a city → rand 0.3747
  - City **B** selected

pij	A	B	C	D	E
B	0.3317	-	0.0000	0.3281	0.3370

- Select a city → rand 0.6216
  - City **D** selected

pij	A	B	C	D	E
D	0.4982	0.0000	0.0000	-	0.4972

- Select a city → rand 0.8033
  - City **E** selected

Ant System
Simple example

- First iteration we can have:
  - Ant #1: C-B-D-E-A
  - Ant #2: D-A-C-B-E
  - Ant #2: A-C-B-D-E
- Update the pheromone using this tours

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

tij	A	B	C	D	E
A	-	0,50	0,54	0,50	0,52
B	0,50	-	0,52	0,52	0,50
C	0,50	0,54	-	0,52	0,52
D	0,54	0,50	0,50	-	0,52
E	0,52	0,52	0,50	0,52	-

- And then iterate

Ant System
Exercise #1

- Implement Ant System according to one of the provided templates.
  - C
  - Java
- The following slides give a practical view of the Ant System algorithm procedures.

Ant System Algorithm
Solution Construction

```
1  Procedure ConstructSolutions
2     For k = 1 To m Do          #m number of ants
3        For i = 1 To n Do       #n number of cities
4           ant[k].visited[i] ← false
5        EndFor
6     EndFor
7     step ← 1
8     For k = 1 To m Do
9        r ← random{1, . . . , n}
10       ant[k].tour [step] ← r
11       ant[k].visited [r] ← true
12    EndFor
```


Ant System Algorithm
Solution Construction

```
13    While (step < n) Do
14       step ← step + 1
15       For k = 1 To m Do
16          ASDecisionRule(k, step)
17       EndFor
18    EndWhile
19    For k = 1 To m Do
20       ant[k].tour [n+1] ← ant[k].tour[1]
21       ant[k].tour length ← ComputeTourLength(k)
22    EndFor
23 EndProcedure
```


Ant System Algorithm
Decision Rule

```
1  Procedure ASDecisionRule(k, i)
2   Input k % ant identifier
3   Input i % counter for construction step
4     c ← ant[k].tour[i-1]
5     sum_prob = 0.0
6     For j = 1 To n Do
7        If ant[k].visited[j] Then
8           selection_prob[j] ← 0.0
9        Else
10          selection_prob[j] ← choice_info[c][j]
11          sum_prob ← sum_prob + selection_prob[j]
12       EndIf
13    EndFor
```


Ant System Algorithm
Decision Rule

```
14    r ← random[0, sum_prob]
15    j ←1
16    p ← selection_prob[j]
17    While (p < r ) Do
18       j ← j + 1
19       p ← p + selection_prob[j]
20    EndWhile
21    ant[k].tour[i] ← j
22    ant[k].visited[j] ← true
23 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 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   Input 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
    - α, β, ρ
- What effect can you appreciate?
- What is the reason?