Swarm Intelligence

Traveling Salesman Problem and Ant System

Leslie Pérez Cáceres
leslie.perez.caceres@ulb.ac.be
Hayfa Hammami
haifa.hammami@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
   - Applied 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

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)

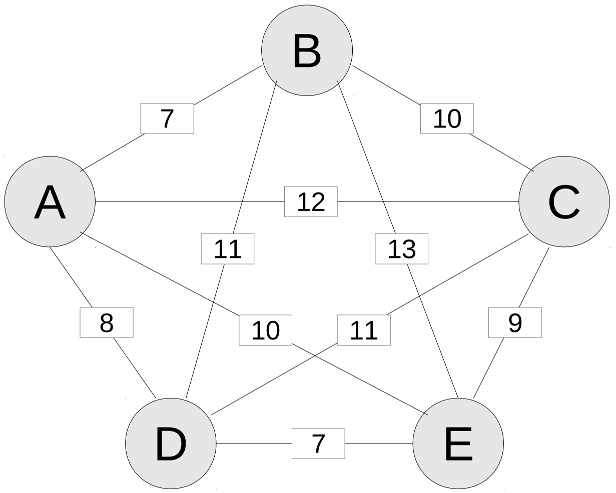
Main reasons for choosing the TSP:

- 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 modelled 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

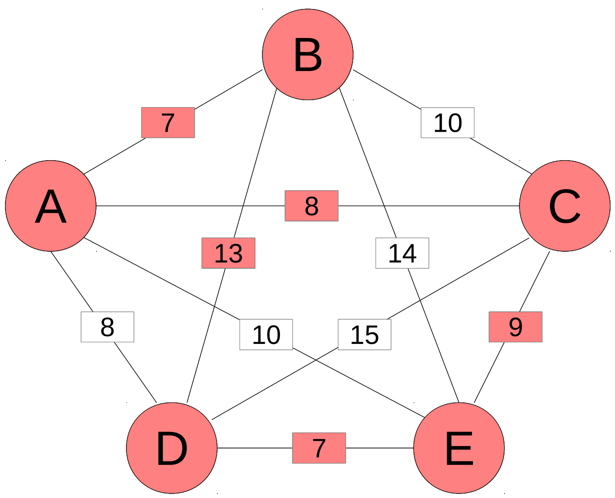
Find a minimum length ***f(π)*** *Hamiltonian circuit of a graph G(N,A), where n is number of nodes and* ***π*** *is a permutation of the nodes indices.*

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

Traveling Tournament Problem

First attempt to solve – Constructive heuristic

- The **nearest neighborhood 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
- 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

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) $\boldsymbol{\eta_{ij}}$
  - Pheromone that represents the learned desirability to visit city ***i*** when in city ***j*** associated to a dynamic value $\boldsymbol{\tau_{ij}}$

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 $\boldsymbol{\tau_{ij}}$
  - heuristic information $\boldsymbol{\eta_{ij} = 1/d_{ij}}$
- Random proportional rule at node ***i*** is:

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

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

Ant System

Pheromone Update

- Example of pheromone update

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

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

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

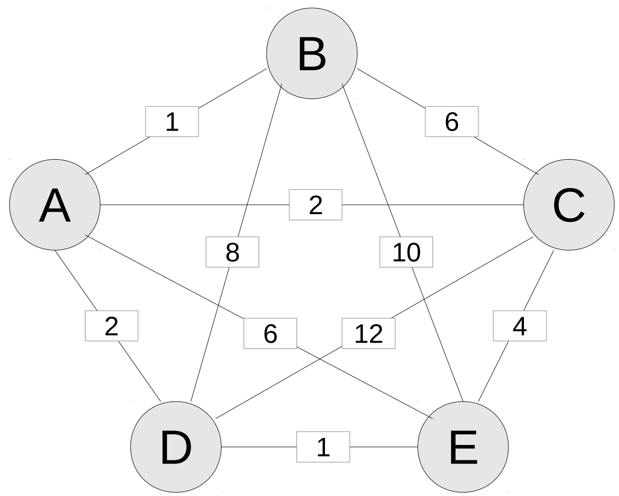
Ant System

Simple pseudo code

```
While !termination()
  For k = 1 To m Do #m number of ants
     ants[k][1] ← SelectRandomCity()
    For i = 2 To n Do #n number of cities
      ants[k][i] ← ASDecisionRule(ants, i)
     EndFor
     ants[k][n+1] ← ants[k][1] #to complete the tour
   EndFor
  UpdatePheromone(ants)
EndWhile
```

Ant System
Simple example

- For our example with #ants=3, α=2, β=1, ρ=0.5 and τ_0=1



  - Heuristic Information

nij	A	B	C	D	E
A	-	1/1	½	½	1/6
B	1/1	-	1/6	1/8	1/10
C	½	1/6	-	1/12	¼
D	½	1/8	1/12	-	1/1
E	1/6	1/10	¼	1/1	-

  - Pheromone trails

tij	A	B	C	D	E
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	-

Ant System

Simple example

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

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

pij	A	B	C	D	E
D	0.264	0.059	0.031	0.000	0.646

[0, 0.264, 0.323, 0.354, 1]

- Select a city → rand 0.80
  - City **E** selected

pij	A	B	C	D	E
E	0.267	0.227	0.506	0.000	0.000

[0, 0.267, 0.494, 1]

- Select a city → rand 0.27
  - City **B** selected

pij	A	B	C	D	E
B	0.843	0.000	0.157	0.000	0.000

[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

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

- And then iterate

tij	A	B	C	D	E
A	-	0.39	0.38	0.42	0.29
B	0.39	-	0.46	0.28	0.35
C	0.38	0.46	-	0.29	0.35
D	0.42	0.28	0.29	-	0.49
E	0.29	0.35	0.35	0.49	-

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                   #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

```
Procedure ASPheromoneUpdate ()
   Evaporate()
   For k = 1 To m Do
      DepositPheromone(k)
   EndFor
   ComputeChoiceInformation()
EndProcedure
```

Ant System Algorithm
Pheromone Update

```
Procedure Evaporate
  For i = 1 To n Do
    For j = i To n Do
      pheromone[i][j] ← (1−ρ)·pheromone[i][j]
      pheromone[j][i] ← pheromone[i][j]
      #pheromones are symmetric
    EndFor
  EndFor
EndProcedure
```

Ant System Algorithm
Pheromone Update

```
Procedure DepositPheromone(k)
   #k ant identifier
   Δτ ← 1/ant[k].tour_length
   For i = 1 To n Do
      j ← ant[k].tour[i]
      l ← ant[k].tour[i+1]
      pheromone[j][l] ← pheromone[j][l] + Δτ
      pheromone[l][j] ← pheromone[j][l]
   EndFor
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?