Metaheuristic Optimization

18. Ant Colony Optimization

Thomas Weise · 汤卫思

tweise@hfuu.edu.cn · http://iao.hfuu.edu.cn

Hefei University, South Campus 2
Faculty of Computer Science and Technology
Institute of Applied Optimization
230601 Shushan District, Hefei, Anhui, China
Econ. & Tech. Devel. Zone, Jinxiu Dadao 99

合肥学院 南艳湖校区/南2区
计算机科学与技术系
应用优化研究所
中国 安徽省 合肥市 蜀山区 230601
经济技术开发区 锦绣大道99号

Outline

1. Introduction
2. Ant Colony Optimization

website

Section Outline

1. Introduction
2. Ant Colony Optimization

- Research by Deneubourg et al. [1–3] on real ants and the simulations by Stickland et al. [4]

Clever Ants

- Research by Deneubourg et al. [1–3] on real ants and the simulations by Stickland et al. [4]
  1. When looking for food, ants move from one location to another

Clever Ants

- Research by Deneubourg et al. [1–3] on real ants and the simulations by Stickland et al. [4]
  1. When looking for food, ants move from one location to another and
  2. lay down pheromone: stigmergy = communication by modifying the environment.

Clever Ants

- Research by Deneubourg et al. [1–3] on real ants and the simulations by Stickland et al. [4]
  1. When looking for food, ants move from one location to another and
  2. lay down pheromone: stigmergy = communication by modifying the environment.
  3. Paths with more pheromone on them are more likely to be followed

Clever Ants

- Research by Deneubourg et al. [1–3] on real ants and the simulations by Stickland et al. [4]
  1. When looking for food, ants move from one location to another and
  2. lay down pheromone: stigmergy = communication by modifying the environment.
  3. Paths with more pheromone on them are more likely to be followed
  4. These are often the shortest paths

- Research by Deneubourg et al. [1–3] on real ants and the simulations by Stickland et al. [4]
  1. When looking for food, ants move from one location to another and
  2. lay down pheromone: stigmergy = communication by modifying the environment.
  3. Paths with more pheromone on them are more likely to be followed
  4. These are often the shortest paths
  5. Many combinatorial problems can be considered as finding the shortest path on a graph. Example: Traveling Salesman Problem

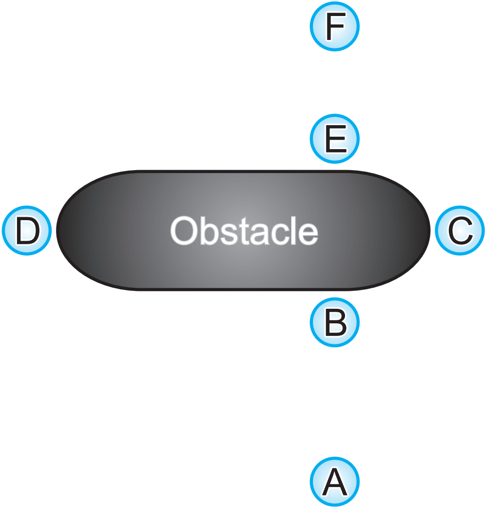
Clever Ants

- Research by Deneubourg et al. [1–3] on real ants and the simulations by Stickland et al. [4]
  1. When looking for food, ants move from one location to another and
  2. lay down pheromone: stigmergy = communication by modifying the environment.
  3. Paths with more pheromone on them are more likely to be followed
  4. These are often the shortest paths
  5. Many combinatorial problems can be considered as finding the shortest path on a graph. Example: Traveling Salesman Problem
- Dorigo et al. [5] have the idea to use a simulation of the way ants form a path in order to solve optimization problems which can be represented as graphs – Ant Colony Optimization (ACO)

Section Outline

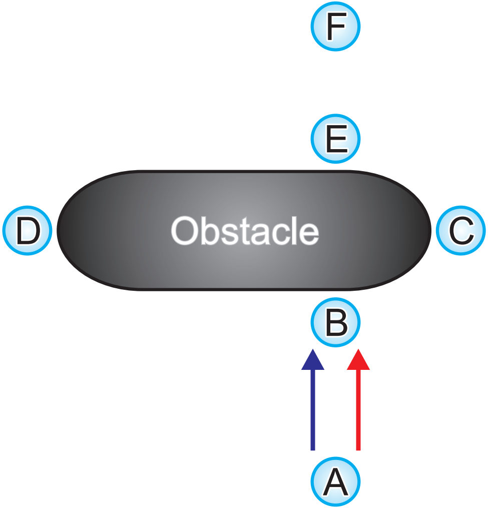
1. Introduction
2. Ant Colony Optimization

- ant nest (A) separated from food source (F) by obstacle



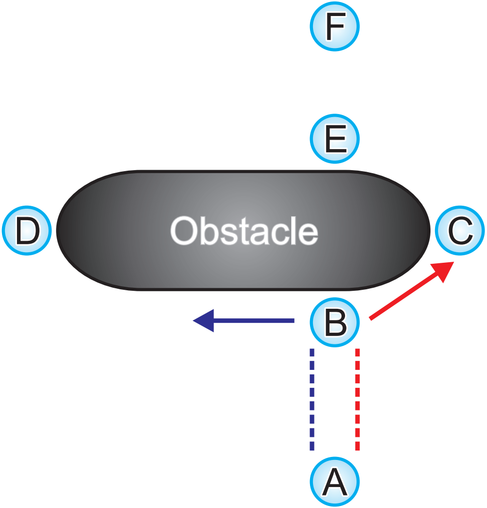
Ant Path Finding

- two ants (red and blue) leave the nest at the same time

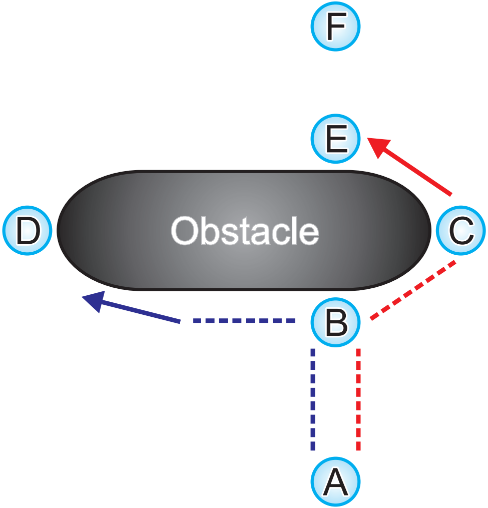


Ant Path Finding

- at the crossroad, one turns left and the other one right

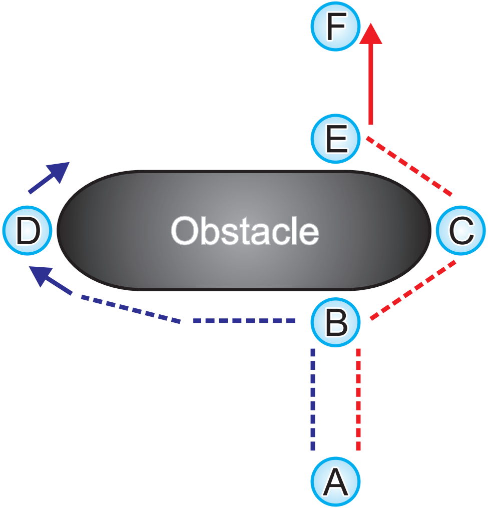


- when moving, ants leave pheromone behind (dotted lines)

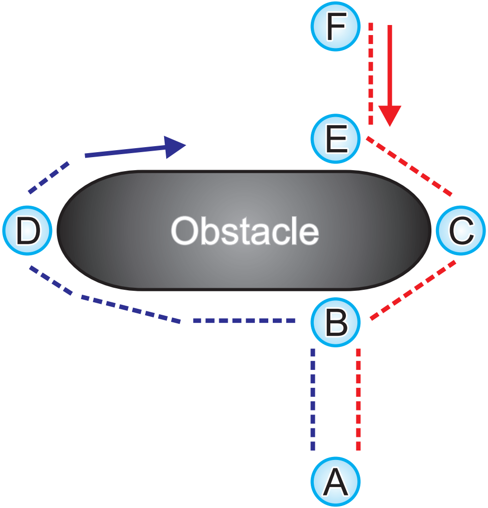


Ant Path Finding

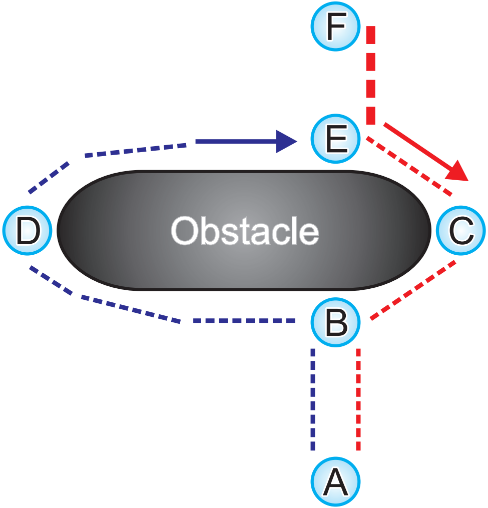
- the one with the shorter path arrives at the food source first



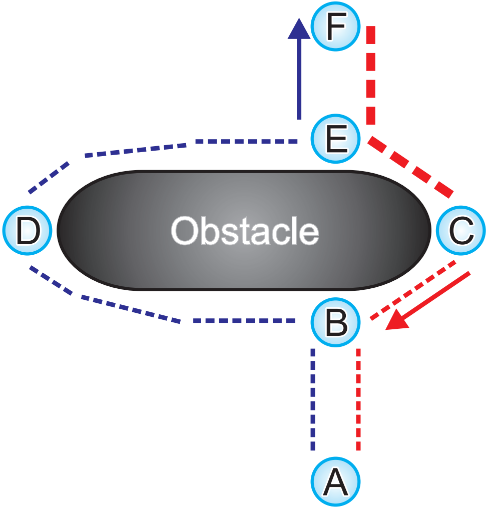
- when it turns back, it finds pheromone on one path and follows it



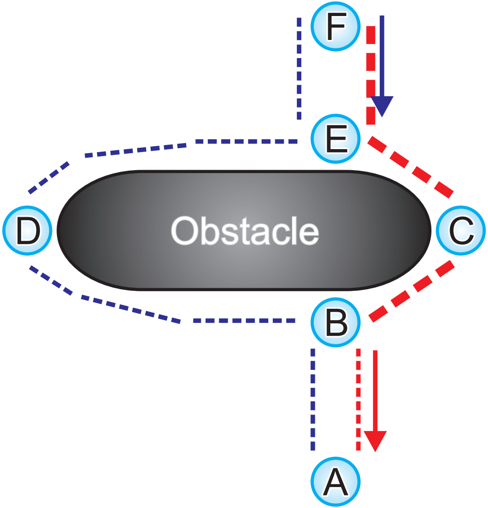
- by doing so, it leaves even more pheromone on the path



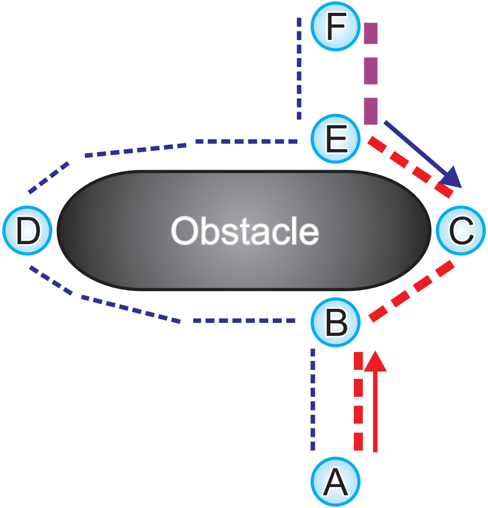
- now the second ant arrives at the food source



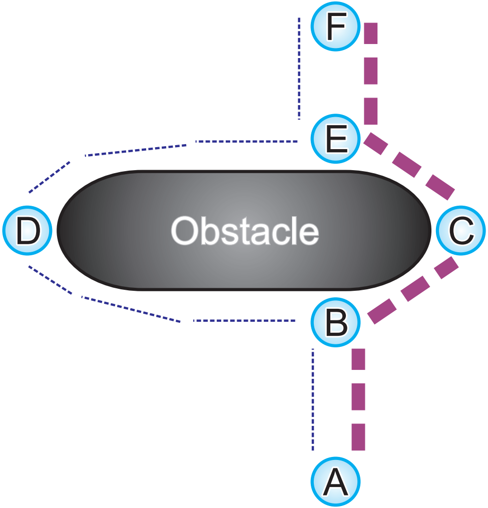
- when it turns back, there is pheromone on both paths – but more on the red one



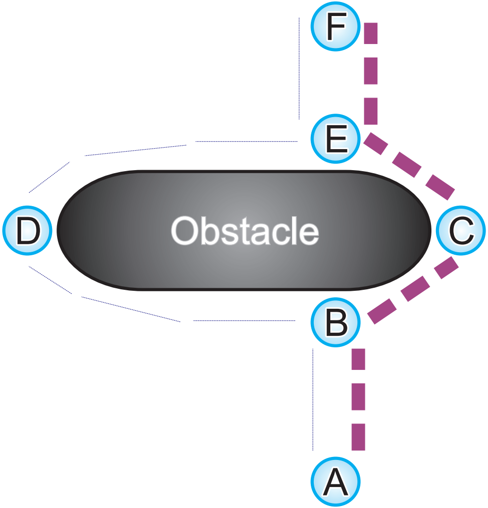
- the pheromone on the short path gets more and more



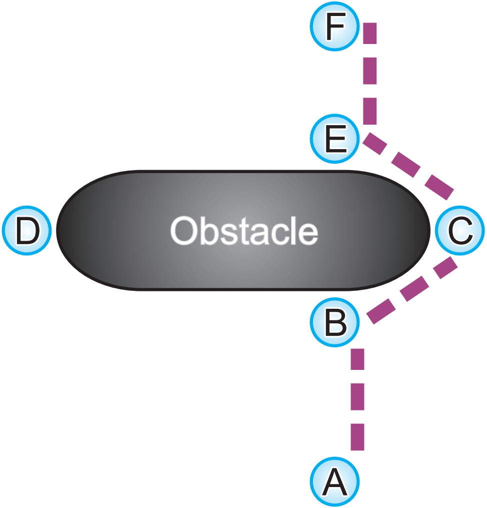
- the pheromone on the short path gets more and more



- while the one on the blue path evaporates



- until only the short path has pheromone. . .



ACO on a Graph

- Like in the example before, assume: there is a set of points on the ground to which an ant can go

ACO on a Graph

- Like in the example before, assume: there is a set of points on the ground to which an ant can go
- These points together with the “ways” between them form a graph

ACO on a Graph

- Like in the example before, assume: there is a set of points on the ground to which an ant can go
- These points together with the “ways” between them form a graph

ACO on a Graph

- Like in the example before, assume: there is a set of points on the ground to which an ant can go
- These points together with the "ways" between them form a graph
- A graph $G = (V, E)$ consists of a set of vertices (nodes) $v \in V$ and edges $e \in E$, with $E \subseteq V \times V$

ACO on a Graph

- Like in the example before, assume: there is a set of points on the ground to which an ant can go
- These points together with the "ways" between them form a graph
- A graph $G = (V, E)$ consists of a set of vertices (nodes) $v \in V$ and edges $e \in E$, with $E \subseteq V \times V$
- ACO has been designed for problems where we want to find paths through such graphs G

- Like in the example before, assume: there is a set of points on the ground to which an ant can go
- These points together with the "ways" between them form a graph
- A graph $G = (V, E)$ consists of a set of vertices (nodes) $v \in V$ and edges $e \in E$, with $E \subseteq V \times V$
- ACO has been designed for problems where we want to find paths through such graphs G
- It is basically a *model-based search algorithm* [6] (similar to EDAs, see Lesson 19: *Estimation of Distribution Algorithms*)

ACO on a Graph

- Like in the example before, assume: there is a set of points on the ground to which an ant can go
- These points together with the "ways" between them form a graph
- A graph $G = (V, E)$ consists of a set of vertices (nodes) $v \in V$ and edges $e \in E$, with $E \subseteq V \times V$
- ACO has been designed for problems where we want to find paths through such graphs G
- It is basically a *model-based search algorithm* [6] (similar to EDAs, see Lesson 19: *Estimation of Distribution Algorithms*):
  - It has a pheromone model τ assigning a pheromone value to each edge $\overline{ij}$

ACO on a Graph

- Like in the example before, assume: there is a set of points on the ground to which an ant can go
- These points together with the "ways" between them form a graph
- A graph $G = (V, E)$ consists of a set of vertices (nodes) $v \in V$ and edges $e \in E$, with $E \subseteq V \times V$
- ACO has been designed for problems where we want to find paths through such graphs G
- It is basically a *model-based search algorithm* [6] (similar to EDAs, see Lesson 19: *Estimation of Distribution Algorithms*):
  - It has a pheromone model τ assigning a pheromone value to each edge $\overline{ij}$
  - τ is sampled, i.e., used to generate new ps paths (1 per ant) through the graph G

ACO on a Graph

- Like in the example before, assume: there is a set of points on the ground to which an ant can go
- These points together with the "ways" between them form a graph
- A graph $G = (V, E)$ consists of a set of vertices (nodes) $v \in V$ and edges $e \in E$, with $E \subseteq V \times V$
- ACO has been designed for problems where we want to find paths through such graphs G
- It is basically a *model-based search algorithm* [6] (similar to EDAs, see Lesson 19: *Estimation of Distribution Algorithms*):
    - It has a pheromone model τ assigning a pheromone value to each edge $\overline{ij}$
    - τ is sampled, i.e., used to generate new ps paths (1 per ant) through the graph G
    - based on the objective value of the resulting paths, τ is updated

ACO on a Graph I

- ACO has three main components

ACO on a Graph I

- ACO has three main components:
  - (simulated) ants which move through a graph along edges.

- ACO has three main components:
  - (simulated) ants which move through a graph along edges. The path such an ant took represents a solution.

ACO on a Graph I

- ACO has three main components:
  - (simulated) ants which move through a graph along edges. The path such an ant took represents a solution.
  - Ants leave pheromones τ on the edges they travel along.

ACO on a Graph I

- ACO has three main components:
  - (simulated) ants which move through a graph along edges. The path such an ant took represents a solution.
  - Ants leave pheromones τ on the edges they travel along. This pheromone helps future ants to decide which path to take.

ACO on a Graph I

- ACO has three main components:
  - (simulated) ants which move through a graph along edges. The path such an ant took represents a solution.
  - Ants leave pheromones τ on the edges they travel along. This pheromone helps future ants to decide which path to take. Pheromone disappears over time (evaporation).

ACO on a Graph I

- ACO has three main components:
  - (simulated) ants which move through a graph along edges. The path such an ant took represents a solution.
  - Ants leave pheromones τ on the edges they travel along. This pheromone helps future ants to decide which path to take. Pheromone disappears over time (evaporation).
  - Knowledge about the problem may be incorporated as a heuristic η which tells the ant how interesting a given edge is.

- ACO has three main components:
  - (simulated) ants which move through a graph along edges. The path such an ant took represents a solution.
  - Ants leave pheromones τ on the edges they travel along. This pheromone helps future ants to decide which path to take. Pheromone disappears over time (evaporation).
  - Knowledge about the problem may be incorporated as a heuristic η which tells the ant how interesting a given edge is. Together with the pheromones, η helps the ant to decide where to go. They don't change over time.

ACO on a Graph II

- Let us assume that all nodes i and $j \in V$ are connected with edges, i.e., we have a complete graph topology

ACO on a Graph II

- Let us assume that all nodes i and $j \in V$ are connected with edges, i.e., we have a complete graph topology
- An ant located in node i in ACO chooses the next node j where it will go according to

- Let us assume that all nodes i and $j \in V$ are connected with edges, i.e., we have a complete graph topology
- An ant located in node i in ACO chooses the next node j where it will go according to
  1. the distance between i and j, and

ACO on a Graph II

- Let us assume that all nodes i and $j \in V$ are connected with edges, i.e., we have a complete graph topology
- An ant located in node i in ACO chooses the next node j where it will go according to
  1. the distance between i and j, and
  2. the amount of pheromone on the edge connecting i and j

- Let us assume that all nodes i and $j \in V$ are connected with edges, i.e., we have a complete graph topology
- An ant located in node i in ACO chooses the next node j where it will go according to
  1. the distance between i and j, and
  2. the amount of pheromone on the edge connecting i and j

$$p_{i,j} = \frac{\left(\tau_{i,j}\right)^{\alpha} * \left(\eta_{i,j}\right)^{\beta}}{\sum_{\forall k} \left(\tau_{i,k}\right)^{\alpha} * \left(\eta_{i,k}\right)} \tag{1}$$

$p_{i,j}$	probability of an ant to go to j if at location i	$\tau_{i,j}$	amount of pheromone on the edge connecting i and j
α, β	weight parameters	$\eta_{i,j}$	visibility of node j from i: inversely proportional to distance between j and i

- At the end of each algorithm round, "pheromone" is dispersed and the trails are updated ($\eta_{i,j}$ stays constant)

- At the end of each algorithm round, “pheromone” is dispersed and the trails are updated ($\eta_{i,j}$ stays constant)

$$\tau_{i,j} = (1-\rho)\tau_{i,j} + \Delta\tau_{i,j} \tag{2}$$

  - ρ is the evaporation coefficient (fraction of pheromone disappearing into thin air)
  - $\Delta\tau_{i,j}$ is the amount of new pheromone dispersed

ACO on a Graph

- At the end of each algorithm round, “pheromone” is dispersed and the trails are updated ($\eta_{i,j}$ stays constant)

$$\tau_{i,j} = (1-\rho)\tau_{i,j} + \Delta\tau_{i,j} \tag{2}$$

  - ρ is the evaporation coefficient (fraction of pheromone disappearing into thin air)
  - $\Delta\tau_{i,j}$ is the amount of new pheromone dispersed
- the amount $\Delta\tau_{i,j}$ usually depends on the quality of the paths the edge (i,j) was part of

A salesman wants to visit n cities in the shortest possible time. No city should be visited twice and he wants arrive back at the origin by the end of the tour.

w(a,b)	Shanghai	Hefei	Guangzhou	Chengdu
Beijing	1244 km	1044 km	2174 km	1854 km
Chengdu	2095 km	1615 km	1954 km	
Guangzhou	1529 km	1257 km		
Hefei	472 km			

Example: Traveling Salesman Problem

A salesman wants to visit n cities in the shortest possible time. No city should be visited twice and he wants arrive back at the origin by the end of the tour.

w(a,b)	Shanghai	Hefei	Guangzhou	Chengdu
Beijing	1244 km	1044 km	2174 km	1854 km
Chengdu	2095 km	1615 km	1954 km	
Guangzhou	1529 km	1257 km		
Hefei	472 km			

- Solution Space:

Example: Traveling Salesman Problem

A salesman wants to visit n cities in the shortest possible time. No city should be visited twice and he wants arrive back at the origin by the end of the tour.

w(a,b)	Shanghai	Hefei	Guangzhou	Chengdu
Beijing	1244 km	1044 km	2174 km	1854 km
Chengdu	2095 km	1615 km	1954 km	
Guangzhou	1529 km	1257 km		
Hefei	472 km			

- Solution Space: $\mathbb{X} = \mathbf{\Pi}\{\text{Beijing}, \text{Chengdu}, \text{Guangzhou}, \text{Hefei}, \text{Shanghai}\}$

A salesman wants to visit n cities in the shortest possible time. No city should be visited twice and he wants arrive back at the origin by the end of the tour.

w(a,b)	Shanghai	Hefei	Guangzhou	Chengdu
Beijing	1244 km	1044 km	2174 km	1854 km
Chengdu	2095 km	1615 km	1954 km	
Guangzhou	1529 km	1257 km		
Hefei	472 km			

- Solution Space: $\mathbb{X} = \mathbf{\Pi}\{\text{Beijing}, \text{Chengdu}, \text{Guangzhou}, \text{Hefei}, \text{Shanghai}\}$
- Objective Function:

A salesman wants to visit n cities in the shortest possible time. No city should be visited twice and he wants arrive back at the origin by the end of the tour.

w(a,b)	Shanghai	Hefei	Guangzhou	Chengdu
Beijing	1244 km	1044 km	2174 km	1854 km
Chengdu	2095 km	1615 km	1954 km	
Guangzhou	1529 km	1257 km		
Hefei	472 km			

- Solution Space: $\mathbb{X} = \mathbf{\Pi}\{\text{Beijing}, \text{Chengdu}, \text{Guangzhou}, \text{Hefei}, \text{Shanghai}\}$
- Objective Function: Minimize $f(x) = \sum_{i=0}^{4} dist(x[i], x[i+1]) + dist(x[4], x[0])$

Example: Traveling Salesman Problem (Idea)

- A TSP is a graph problem by default

Example: Traveling Salesman Problem (Idea)

- A TSP is a graph problem by default
- We look for a path that visits all n nodes in a graph

- A TSP is a graph problem by default
- We look for a path that visits all n nodes in a graph (the return back to the start can be added automatically)

Example: Traveling Salesman Problem (Idea)

- A TSP is a graph problem by default
- We look for a path that visits all n nodes in a graph (the return back to the start can be added automatically)
- Let's apply ACO!

Example: Traveling Salesman Problem (Idea)

- A TSP is a graph problem by default
- We look for a path that visits all n nodes in a graph (the return back to the start can be added automatically)
- Let's apply ACO!
- Basic Idea

Example: Traveling Salesman Problem (Idea)

- A TSP is a graph problem by default
- We look for a path that visits all n nodes in a graph (the return back to the start can be added automatically)
- Let's apply ACO!
- Basic Idea:
  - the cities are connected with edges

Example: Traveling Salesman Problem (Idea)

- A TSP is a graph problem by default
- We look for a path that visits all n nodes in a graph (the return back to the start can be added automatically)
- Let's apply ACO!
- Basic Idea:
  - the cities are connected with edges
  - we have ps ants

Example: Traveling Salesman Problem (Idea)

- A TSP is a graph problem by default
- We look for a path that visits all n nodes in a graph (the return back to the start can be added automatically)
- Let's apply ACO!
- Basic Idea:
  - the cities are connected with edges
  - we have ps ants
  - ant k moves from one city to one of the cities it has not seen yet based on a given probability

Example: Traveling Salesman Problem (Idea)

- A TSP is a graph problem by default
- We look for a path that visits all n nodes in a graph (the return back to the start can be added automatically)
- Let's apply ACO!
- Basic Idea:
  - the cities are connected with edges
  - we have ps ants
  - ant k moves from one city to one of the cities it has not seen yet based on a given probability
  - this probability depends on the pheromones on the edges and the distances to the cities

Example: Traveling Salesman Problem (Idea)

- A TSP is a graph problem by default
- We look for a path that visits all n nodes in a graph (the return back to the start can be added automatically)
- Let's apply ACO!
- Basic Idea:
  - the cities are connected with edges
  - we have ps ants
  - ant k moves from one city to one of the cities it has not seen yet based on a given probability
  - this probability depends on the pheromones on the edges and the distances to the cities
  - after all ants have completed their tour, pheromones are updated

Example: Traveling Salesman Problem

In each iteration of the ACO do

In each iteration of the ACO do:

1. For each ant k of the ps ants

Example: Traveling Salesman Problem

In each iteration of the ACO do:

1. For each ant k of the ps ants:
    1. Place ant k at a randomly chosen city/node i

In each iteration of the ACO do:

1. For each ant k of the ps ants:
    1. Place ant k at a randomly chosen city/node i
    2. For $n-1$ times

Example: Traveling Salesman Problem

In each iteration of the ACO do:

1. For each ant k of the ps ants:
   1. Place ant k at a randomly chosen city/node i
   2. For $n-1$ times:
      1. Choose next city j from the set of cities not yet visited by the ant (where i is its current location)

Example: Traveling Salesman Problem

In each iteration of the ACO do:

1. For each ant k of the ps ants:
   1. Place ant k at a randomly chosen city/node i
   2. For $n-1$ times:
      1. Choose next city j from the set of cities not yet visited by the ant (where i is its current location)
      2. j has probability $p_{i,j}$ to be chosen as next city:

$$p_{i,j} \propto (\tau_{i,j})^{\alpha} * (\eta_{i,j})^{\beta} \tag{3}$$

where $\tau_{i,j}$ is the pheromone and $\eta_{i,j} = \frac{1}{dist(i,j)}$, and α, β are weight parameters

Example: Traveling Salesman Problem

In each iteration of the ACO do:

1. For each ant k of the ps ants:
   1. Place ant k at a randomly chosen city/node i
   2. For $n-1$ times:
      1. Choose next city j from the set of cities not yet visited by the ant (where i is its current location)
      2. j has probability $p_{i,j}$ to be chosen as next city:

$$p_{i,j} = \frac{\left(\tau_{i,j}\right)^{\alpha} * \left(\eta_{i,j}\right)^{\beta}}{\sum_{\forall k} \left(\tau_{i,k}\right)^{\alpha} * \left(\eta_{i,k}\right)} \tag{3}$$

where $\tau_{i,j}$ is the pheromone and $\eta_{i,j} = \frac{1}{dist(i,j)}$, and α, β are weight parameters

Example: Traveling Salesman Problem

In each iteration of the ACO do:

1. For each ant k of the ps ants:
   1. Place ant k at a randomly chosen city/node i
   2. For $n-1$ times:
      1. Choose next city j from the set of cities not yet visited by the ant (where i is its current location)
      2. j has probability $p_{i,j}$ to be chosen as next city:

$$p_{i,j} = \frac{\left(\tau_{i,j}\right)^{\alpha} * \left(\eta_{i,j}\right)^{\beta}}{\sum_{\forall k}\left(\tau_{i,k}\right)^{\alpha} * \left(\eta_{i,k}\right)} \tag{3}$$

      where $\tau_{i,j}$ is the pheromone and $\eta_{i,j} = \frac{1}{dist(i,j)}$, and α, β are weight parameters
   3. Add the trip back to the starting node

In each iteration of the ACO do:

1. For each ant k of the ps ants:
    1. Place ant k at a randomly chosen city/node i
    2. For $n-1$ times:
        1. Choose next city j from the set of cities not yet visited by the ant (where i is its current location)
        2. j has probability $p_{i,j}$ to be chosen as next city:

$$p_{i,j} = \frac{(\tau_{i,j})^{\alpha} * (\eta_{i,j})^{\beta}}{\sum_{\forall k} (\tau_{i,k})^{\alpha} * (\eta_{i,k})} \tag{3}$$

where $\tau_{i,j}$ is the pheromone and $\eta_{i,j} = \frac{1}{dist(i,j)}$, and α, β are weight parameters

    3. Add the trip back to the starting node $\Longrightarrow$ We get tour x_k

In each iteration of the ACO do:

1. For each ant k of the ps ants:
    1. Place ant k at a randomly chosen city/node i
    2. For $n-1$ times:
        1. Choose next city j from the set of cities not yet visited by the ant (where i is its current location)
        2. j has probability $p_{i,j}$ to be chosen as next city $p_{i,j} \propto (\tau_{i,j})^{\alpha} * (\eta_{i,j})^{\beta}$ where $\tau_{i,j}$ is the pheromone and $\eta_{i,j} = \frac{1}{dist(i,j)}$, and α, β are weight parameters
    3. Add the trip back to the starting node $\Longrightarrow$ We get tour x_k

In each iteration of the ACO do:

1. For each ant k of the ps ants:
   1. Place ant k at a randomly chosen city/node i
   2. For $n-1$ times:
      1. Choose next city j from the set of cities not yet visited by the ant (where i is its current location)
      2. j has probability $p_{i,j}$ to be chosen as next city $p_{i,j} \propto (\tau_{i,j})^{\alpha} * (\eta_{i,j})^{\beta}$ where $\tau_{i,j}$ is the pheromone and $\eta_{i,j} = \frac{1}{dist(i,j)}$, and α, β are weight parameters
   3. Add the trip back to the starting node $\Longrightarrow$ We get tour x_k
2. Calculate pheromone amount $\Delta\tau_{i,j}$ to be dispersed on the edge $\overline{ij}$ connecting city i with city j

Example: Traveling Salesman Problem

In each iteration of the ACO do:

1. For each ant k of the ps ants:
   1. Place ant k at a randomly chosen city/node i
   2. For $n-1$ times:
      1. Choose next city j from the set of cities not yet visited by the ant (where i is its current location)
      2. j has probability $p_{i,j}$ to be chosen as next city $p_{i,j} \propto (\tau_{i,j})^{\alpha} * (\eta_{i,j})^{\beta}$ where $\tau_{i,j}$ is the pheromone and $\eta_{i,j} = \frac{1}{dist(i,j)}$, and α, β are weight parameters
   3. Add the trip back to the starting node $\Longrightarrow$ We get tour x_k
2. Calculate pheromone amount $\Delta\tau_{i,j}$ to be dispersed on the edge $\overline{ij}$ connecting city i with city j as:

$$\sum_{k=1}^{ps} \begin{cases} \frac{1}{f(x_k)} & \text{if tour } x_k \text{ contains edge } \overline{ij} \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

Example: Traveling Salesman Problem

In each iteration of the ACO do:

1. For each ant k of the ps ants:
   1. Place ant k at a randomly chosen city/node i
   2. For $n-1$ times:
      1. Choose next city j from the set of cities not yet visited by the ant (where i is its current location)
      2. j has probability $p_{i,j}$ to be chosen as next city $p_{i,j} \propto (\tau_{i,j})^{\alpha} * (\eta_{i,j})^{\beta}$ where $\tau_{i,j}$ is the pheromone and $\eta_{i,j} = \frac{1}{dist(i,j)}$, and α, β are weight parameters
   3. Add the trip back to the starting node $\Longrightarrow$ We get tour x_k
2. Calculate pheromone amount $\Delta\tau_{i,j}$ to be dispersed on the edge $\overline{ij}$ connecting city i with city j as:

$$\sum_{k=1}^{ps} \begin{cases} \frac{1}{f(x_k)} & \text{if tour } x_k \text{ contains edge } \overline{ij} \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

   Alternative method: only the best ant contributes to $\Delta\tau$

Example: Traveling Salesman Problem

In each iteration of the ACO do:

1. For each ant k of the ps ants:
   1. Place ant k at a randomly chosen city/node i
   2. For $n-1$ times:
      1. Choose next city j from the set of cities not yet visited by the ant (where i is its current location)
      2. j has probability $p_{i,j}$ to be chosen as next city $p_{i,j} \propto (\tau_{i,j})^{\alpha} * (\eta_{i,j})^{\beta}$ where $\tau_{i,j}$ is the pheromone and $\eta_{i,j} = \frac{1}{dist(i,j)}$, and α, β are weight parameters
   3. Add the trip back to the starting node $\Longrightarrow$ We get tour x_k
2. Calculate pheromone amount $\Delta\tau_{i,j}$ to be dispersed on the edge $\overline{ij}$
3. Update pheromone value $\tau_{i,j}$

In each iteration of the ACO do:

1. For each ant k of the ps ants:
   1. Place ant k at a randomly chosen city/node i
   2. For $n-1$ times:
      1. Choose next city j from the set of cities not yet visited by the ant (where i is its current location)
      2. j has probability $p_{i,j}$ to be chosen as next city $p_{i,j} \propto (\tau_{i,j})^{\alpha} * (\eta_{i,j})^{\beta}$ where $\tau_{i,j}$ is the pheromone and $\eta_{i,j} = \frac{1}{dist(i,j)}$, and α, β are weight parameters
   3. Add the trip back to the starting node $\Longrightarrow$ We get tour x_k
2. Calculate pheromone amount $\Delta\tau_{i,j}$ to be dispersed on the edge $\overline{ij}$
3. Update pheromone value $\tau_{i,j}$ according to

$$\tau_{i,j} = (1-\rho)\tau_{i,j} + \Delta\tau_{i,j} \tag{3}$$

where the evaporation coefficient $\rho \in [0,1]$ lets old pheromone disappear

In each iteration of the ACO do:

1. For each ant k of the ps ants:
    1. Place ant k at a randomly chosen city/node i
    2. For $n-1$ times:
        1. Choose next city j from the set of cities not yet visited by the ant (where i is its current location)
        2. j has probability $p_{i,j}$ to be chosen as next city $p_{i,j} \propto (\tau_{i,j})^{\alpha} * (\eta_{i,j})^{\beta}$ where $\tau_{i,j}$ is the pheromone and $\eta_{i,j} = \frac{1}{dist(i,j)}$, and α, β are weight parameters
    3. Add the trip back to the starting node $\Longrightarrow$ We get tour x_k
2. Calculate pheromone amount $\Delta\tau_{i,j}$ to be dispersed on the edge $\overline{ij}$
3. Update pheromone value $\tau_{i,j}$

Example: Traveling Salesman Problem

In each iteration of the ACO do:

1. For each ant k of the ps ants:
   1. Place ant k at a randomly chosen city/node i
   2. For $n-1$ times:
      1. Choose next city j from the set of cities not yet visited by the ant (where i is its current location)
      2. j has probability $p_{i,j}$ to be chosen as next city $p_{i,j} \propto (\tau_{i,j})^{\alpha} * (\eta_{i,j})^{\beta}$ where $\tau_{i,j}$ is the pheromone and $\eta_{i,j} = \frac{1}{dist(i,j)}$, and α, β are weight parameters
   3. Add the trip back to the starting node $\Longrightarrow$ We get tour x_k
2. Calculate pheromone amount $\Delta\tau_{i,j}$ to be dispersed on the edge $\overline{ij}$
3. Update pheromone value $\tau_{i,j}$

return the best tour $x^{\star}$ discovered.

In each iteration of the ACO do:

1. For each ant k of the ps ants:
   1. Place ant k at a randomly chosen city/node i
   2. For $n-1$ times:
      1. Choose next city j from the set of cities not yet visited by the ant (where i is its current location)
      2. j has probability $p_{i,j}$ to be chosen as next city $p_{i,j} \propto (\tau_{i,j})^{\alpha} * (\eta_{i,j})^{\beta}$ where $\tau_{i,j}$ is the pheromone and $\eta_{i,j} = \frac{1}{dist(i,j)}$, and α, β are weight parameters
   3. Add the trip back to the starting node $\Longrightarrow$ We get tour x_k
2. Calculate pheromone amount $\Delta\tau_{i,j}$ to be dispersed on the edge $\overline{ij}$
3. Update pheromone value $\tau_{i,j}$

return the best tour $x^{\star}$ discovered.

New Perspective: Path through Graph $\Leftrightarrow$ Permutation

In each iteration of the ACO do:

1. For each ant k of the ps ants:
   1. Place ant k at a randomly chosen city/node i
   2. For $n-1$ times:
      1. Choose next city j from the set of cities not yet visited by the ant (where i is its current location)
      2. j has probability $p_{i,j}$ to be chosen as next city $p_{i,j} \propto (\tau_{i,j})^{\alpha} * (\eta_{i,j})^{\beta}$ where $\tau_{i,j}$ is the pheromone and $\eta_{i,j} = \frac{1}{dist(i,j)}$, and α, β are weight parameters
   3. Add the trip back to the starting node $\Longrightarrow$ We get tour x_k
2. Calculate pheromone amount $\Delta\tau_{i,j}$ to be dispersed on the edge $\overline{ij}$
3. Update pheromone value $\tau_{i,j}$

return the best tour $x^{\star}$ discovered.

New Perspective: Path through Graph $\Leftrightarrow$ Permutation $\Longrightarrow$ ACO is good for permutation-based problems

Pheromones

- One pheromone value $\tau_{i,j}$ for each edge $\overline{i,j}$ in the graph

Pheromones

- One pheromone value $\tau_{i,j}$ for each edge $\overline{i,j}$ in the graph
- If there are n nodes, there may be $\frac{n(n-1)}{2}$ undirected or $n(n-1)$ directed edges

Pheromones

- One pheromone value $\tau_{i,j}$ for each edge $\overline{i,j}$ in the graph
- If there are n nodes, there may be $\frac{n(n-1)}{2}$ undirected or $n(n-1)$ directed edges
- Pheromones τ usually maintained in a matrix data structure with $\mathcal{O}\left(n^2\right)$ elements

Pheromones

- One pheromone value $\tau_{i,j}$ for each edge $\overline{i,j}$ in the graph
- If there are n nodes, there may be $\frac{n(n-1)}{2}$ undirected or $n(n-1)$ directed edges
- Pheromones τ usually maintained in a matrix data structure with $\mathcal{O}\left(n^2\right)$ elements
- High memory consumption $(\mathcal{O}\left(n^2\right))$ and update step of this matrix is also slow $(\mathcal{O}\left(n^2\right))$...

- Idea of *Population-based ACO* [7, 8]

pACO

- Idea of *Population-based ACO* [7, 8]:
  - remember best b ants

- Idea of *Population-based ACO* [7, 8]:
  - remember best b ants
  - only these ants determine pheromone

pACO

- Idea of *Population-based ACO* [7, 8]:
  - remember best b ants
  - only these ants determine pheromone
  - if a better ant is discovered

pACO

- Idea of *Population-based ACO*[7, 8]:
  - remember best b ants
  - only these ants determine pheromone
  - if a better ant is discovered:
    - remove pheromone of worst/oldest ant

- Idea of *Population-based ACO* [7, 8]:
  - remember best b ants
  - only these ants determine pheromone
  - if a better ant is discovered:
    - remove pheromone of worst/oldest ant
    - add pheromone for new ant

- Idea of *Population-based ACO* [7, 8]:
  - remember best b ants
  - only these ants determine pheromone
  - if a better ant is discovered:
    - remove pheromone of worst/oldest ant
    - add pheromone for new ant
  - this needs only $\mathcal{O}(n)$ steps for updates and has a memory consumption of $\mathcal{O}(n)$

- Idea of *Population-based ACO* [7, 8]:
  - remember best b ants
  - only these ants determine pheromone
  - if a better ant is discovered:
    - remove pheromone of worst/oldest ant
    - add pheromone for new ant
  - this needs only $\mathcal{O}(n)$ steps for updates and has a memory consumption of $\mathcal{O}(n)$
  - and provides better results [7]

- Idea of *Population-based ACO* [7, 8]:
  - remember best b ants
  - only these ants determine pheromone
  - if a better ant is discovered:
    - remove pheromone of worst/oldest ant
    - add pheromone for new ant
  - this needs only $\mathcal{O}(n)$ steps for updates and has a memory consumption of $\mathcal{O}(n)$
  - and provides better results [7]
  - and is suitable for dynamically changing problems [8]

Summary

- ACO

Summary

- ACO
  - can solve problems that 1) involve permutations, 2) involve paths through graphs

Summary

- ACO
  - can solve problems that 1) involve permutations, 2) involve paths through graphs
  - Maintaining pheromone matrix τ: can be reduced in complexity to $\mathcal{O}(n)$

Summary

- ACO
  - can solve problems that 1) involve permutations, 2) involve paths through graphs
  - Maintaining pheromone matrix τ: can be reduced in complexity to $\mathcal{O}(n)$
  - Can be applied to dynamic problems

Summary

- ACO
  - can solve problems that 1) involve permutations, 2) involve paths through graphs
  - Maintaining pheromone matrix τ: can be reduced in complexity to $\mathcal{O}(n)$
  - Can be applied to dynamic problems
- Collectives of simple creatures able to colossal achievements

Summary

- ACO
  - can solve problems that 1) involve permutations, 2) involve paths through graphs
  - Maintaining pheromone matrix τ: can be reduced in complexity to $\mathcal{O}(n)$
  - Can be applied to dynamic problems
- Collectives of simple creatures able to colossal achievements:
  - Swarm Intelligence

Summary

- ACO
  - can solve problems that 1) involve permutations, 2) involve paths through graphs
  - Maintaining pheromone matrix τ: can be reduced in complexity to $\mathcal{O}(n)$
  - Can be applied to dynamic problems
- Collectives of simple creatures able to colossal achievements:
  - Swarm Intelligence
  - PSO: Copy flocking behavior

Summary

- ACO
  - can solve problems that 1) involve permutations, 2) involve paths through graphs
  - Maintaining pheromone matrix τ: can be reduced in complexity to $\mathcal{O}(n)$
  - Can be applied to dynamic problems
- Collectives of simple creatures able to colossal achievements:
  - Swarm Intelligence
  - PSO: Copy flocking behavior
  - ACO: Copy ground-based movements / stigmergy

谢谢

Thank you

Thomas Weise [汤卫思]
tweise@hfuu.edu.cn
http://iao.hfuu.edu.cn

Hefei University, South Campus 2
Institute of Applied Optimization
Shushan District, Hefei, Anhui,
China

Caspar David Friedrich, “Der Wanderer über dem Nebelmeer”, 1818
http://en.wikipedia.org/wiki/Wanderer_above_the_Sea_of_Fog

Bibliography

Bibliography I

1. Jean-Louis Deneubourg, Jacques M. Pasteels, and J. C. Verhaeghe. Probabilistic behaviour in ants: A strategy of errors? *Journal of Theoretical Biology*, 105(2):259–271, 1983. doi: 10.1016/S0022-5193(83)80007-1.
2. Jean-Louis Deneubourg
and Simon Goss. Collective patterns and decision-making. *Ethology, Ecology & Evolution*, 1(4):295–311, December 1989. URL
`http://www.ulb.ac.be/sciences/use/publications/JLD/53.pdfhttp://www.ulb.ac.be/sciences/use/publications/JLD/`
3. Simon Goss, R. Beckers, Jean-Louis Deneubourg, S. Aron, and Jacques M. Pasteels. How trail laying and trail following can solve foraging problems for ant colonies. In Roger N. Hughes, editor, *NATO Advanced Research Workshop on Behavioural Mechanisms of Food Selection*, volume 20 of *NATO Advanced Science Institutes (ASI) Series G. Ecological Sciences (NATO ASI)*, pages 661–678, Gregynog, Wales, UK, July 17–21, 1989. Berlin, Germany: Springer-Verlag GmbH.
4. T. R. Stickland, Chris M. N. Tofts, and Nigel R. Franks. A path choice algorithm for ants. *Naturwissenschaften – The Science of Nature*, 79(12):567–572, December 1992. doi: 10.1007/BF01131415.
5. Marco Dorigo, Vittorio Maniezzo, and Alberto Colorni. The ant system: Optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics – Part B: Cybernetics*, 26(1):29–41, February 1996. doi: 10.1109/3477.484436. URL `ftp://iridia.ulb.ac.be/pub/mdorigo/journals/IJ.10-SMC96.pdf`.
6. Mark Zlochin, Mauro Birattari, Nicolas Meuleau, and Marco Dorigo. Model-based search for combinatorial optimization: A critical survey. *Annals of Operations Research*, 132(1-4):373–395, November 2004. doi: 10.1023/B:ANOR.0000039526.52305.af.
7. Michael Guntsch and Martin Middendorf. A population based approach for aco. In Stefano Cagnoni, Jens Gottlieb, Emma Hart, Martin Middendorf, and Günther R. Raidl, editors, *Applications of Evolutionary Computing, Proceedings of EvoWorkshops 2002: EvoCOP, EvoIASP, EvoSTIM/EvoPLAN (EvoWorkshops'02)*, volume 2279 of *Lecture Notes in Computer Science (LNCS)*, pages 72–81, Kinsale, Ireland, April 2–4, 2002. Berlin, Germany: Springer-Verlag GmbH. doi: 10.1007/3-540-46004-7_8. URL `http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.13.2514`.
8. Michael Guntsch and Martin Middendorf. Applying population based aco to dynamic optimization problems. In Marco Dorigo, Gianni A. Di Caro, and Michael Samples, editors, *From Ant Colonies to Artificial Ants – Proceedings of the Third International Workshop on Ant Colony Optimization (ANTS'02)*, volume 2463/2002 of *Lecture Notes in Computer Science (LNCS)*, pages 111–122, Brussels, Belgium, September 12–14, 2002. Berlin, Germany: Springer-Verlag GmbH. doi: 10.1007/3-540-45724-0_10. URL `http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.12.6580`.