Any Colony Optimization

Adam DeJans

Oakland University

addejans@oakland.edu

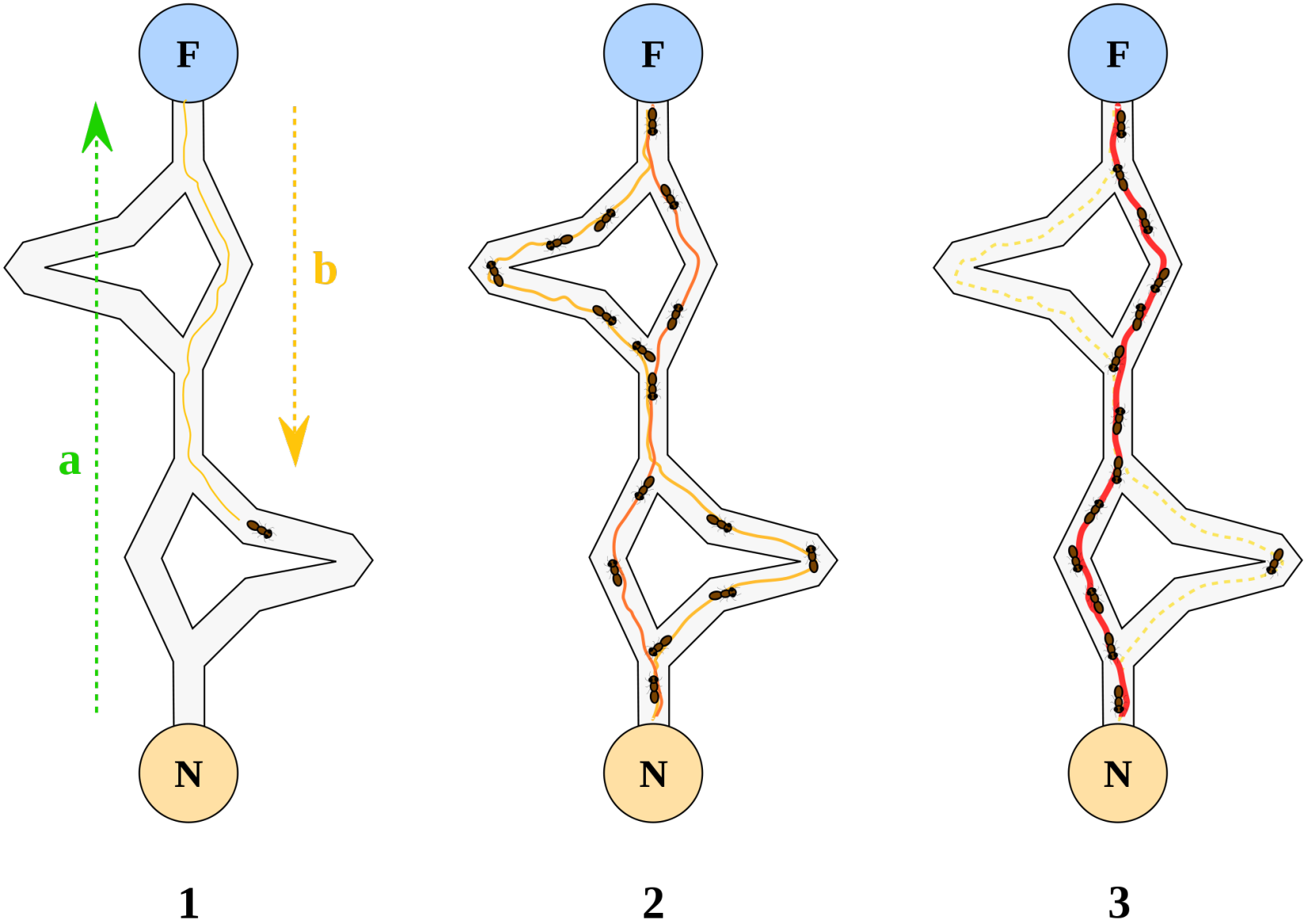
February 22, 2017

Overview

Introduction

Ant algorithms are inspired from the observation of real ant colonies. In particular, the how ants are able to find shortest paths between food sources and their nest.

Pictorial Example



[1]

[1] https://upload.wikimedia.org/wikipedia/commons/thumb/a/af/Aco_branches.svg/2000px-Aco_branches.svg.png

Real vs. Simulated Ants

Similarities

- Colony of cooperating individuals
- Pheromone trail and stigmergy
- Shortest path searching and local moves
- Stochastic and myopic state transition policy

Real vs. Simulated Ants

Similarities

- Colony of cooperating individuals
- Pheromone trail and stigmergy
- Shortest path searching and local moves
- Stochastic and myopic state transition policy

Unique to Artificial Ants:

- live in a discrete world
- have an internal state
- deposit an amount of pheromone which is a function of the quality of the solution found
- in many cases update pheromone trails only after having a generated solution
- have extra capabilities (e.g., looking ahead, local optimization, backtracking, etc.)

Ant System (1996)

"A new general-purpose heuristic algorithm to solve different optimization problems"

Ant System (1996)

- Originally designed to solve the Travelling Salesman Problem (TSP)

Ant System (1996)

- Originally designed to solve the Travelling Salesman Problem (TSP)
- That is; given V cities, and a distance function $d_{i,j}$ between cities i and j, find a tour that visits every city exactly once and minimizes the total distance.

Ant System (1996)

- Originally designed to solve the Travelling Salesman Problem (TSP)
- That is; given V cities, and a distance function $d_{i,j}$ between cities i and j, find a tour that visits every city exactly once and minimizes the total distance.
- This is a classical NP-hard combinatorial optimization problem.

Ant System (1996)

- Originally designed to solve the Travelling Salesman Problem (TSP)
- That is; given V cities, and a distance function $d_{i,j}$ between cities i and j, find a tour that visits every city exactly once and minimizes the total distance.
- This is a classical NP-hard combinatorial optimization problem.
- TSP is a very important problem in regards to Ant Colony Optimization (ACO) because the original ACO algorithm, Ant System (AS), was first applied to TSP and TSP is often used as a benchmark to test a new algorithmic ideas.

AS Pseudocode for TSP

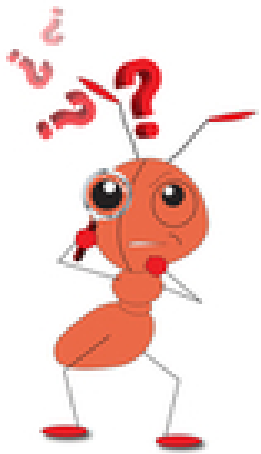
```
def AS_TSP():
    initialize()
    random_ant_placement()
    while (stopping_criteria == False):
        for (ant in Ants):
            while (more_cities_toVisit == True):
                transition_ant()
            return_to_initial_city()
            pheromone_update()
```

Tour Construction

- m (artificial) ants are created to concurrently build a tour of the TSP
- Ants are initially put on randomly chosen cities.

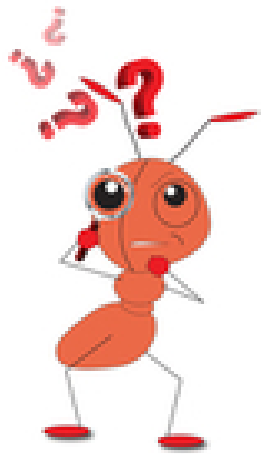
Path Selection

How does an artificial ant choose which vertex to visit next?



Path Selection

How does an artificial ant choose which vertex to visit next?



This is determined probabilistically

Path Selection - Transition Probability

The transition probability from city i to city j for the k-th ant is defined as

$$p_{i,j}^k(t) = \begin{cases} \frac{[\tau_{i,j}(t)]^\alpha \cdot [\eta_{i,j}(t)]^\beta}{\sum_{l \in \mathcal{N}_i^k} [\tau_{i,l}(t)]^\alpha \cdot [\eta_{i,l}(t)]^\beta} & \text{if } j \in \mathcal{N}_i^k \\ 0 & \text{otherwise} \end{cases}$$

[2]

Variables:

- $\tau_{i,j}(t) =$ amount of pheromone trail on edge (i,j) at time t.
- $\eta_{i,j} =$ the visibility; expressed as $\frac{1}{d_{i,j}}$, where $d_{i,j}$ is the euclidean distance between two vertices.
- $\alpha =$ parameter controlling the relative weight of pheromone trail
- $\beta =$ parameter controlling the weight of visibility

[2] Notice: the fraction ensures that $\sum_{j \in V} p_{i,j}^k(t) = 1$

Path Selection - Transition Probability (cont.)

Data Structures:

- $tabu_k$ – dynamically growing vector containing the tabu list of the k-th ant (list of vertices visited by the k-th ant)
- $\mathcal{N}_i^k = V \setminus tabu_k$ – list of vertices not visited by the k-th ant

Updating Pheromone

After each ant has completed their tour, pheromone evaporation on all arcs is triggered, and then each ant k deposits a quantity of pheromone $\triangle\tau_{i,j}^{k}(t)$ on each arc that it has used.

[3] was found to have negligible influence

Updating Pheromone

After each ant has completed their tour, pheromone evaporation on all arcs is triggered, and then each ant k deposits a quantity of pheromone $\triangle\tau_{i,j}^k(t)$ on each arc that it has used.

$$\triangle\tau_{i,j}^k(t) = \begin{cases} \frac{Q}{L^k(t)} & \text{if arc } (i,j) \in tabu_k(t) \\ 0 & \text{otherwise} \end{cases}$$

Q is a constant[3]
$L^k(t)$ is the length of the tour done by ant k at iteration t

[3] was found to have negligible influence

Updating Pheromone (cont.)

One can implement the addition of new pheromone and pheromone evaporation to all arcs by

$$\tau_{i,j}(t) \leftarrow (1-\rho)\tau_{i,j}(t) + \triangle\tau_{i,j}(t)$$

where $\triangle\tau_{i,j}(t) = \sum_{k=1}^{m} \triangle\tau_{i,j}^{k}(t)$ and ρ is a constant in $(0, 1]$ representing *evaporation*.

End of First Run

1. Save best ant tour (sequence and length)
2. All ants die
3. New ants are born

Stopping Criteria

The algorithm can be set to stop after

- Stagnation; or
- A maximum number of iterations is reached

Runtime

- Not so great results
- Modified to now have excellent results (Ant Colony System)
- *daemon_actions*() - the optional function used to implement centralized actions that can't be performed by individual ants.
  - E.g., using global perspective to decide whether or not to deposit additional pheromone to persuade the search process from a non-local aspect.
  - Local search methods to constructed solutions

Ant Colony System (ACS)

Three major modifications to enhance results:

1. Search experience accumulated from ants is used more strongly
2. Pheromone evaporation and deposit take place only on arcs belonging to the best-so-far tour
3. Each time an ant uses an arc it removes some pheromone from the arc to increase the exploration of alternative paths.

ACS Pseudocode for TSP

```
def ACS_TSP():
    while (stopping_criteria == False):
        for (city in Cities):
            for (ant in Ants)
                transition_ant()
                local_trail_update()
        global_trail_update
```

Tour Construction (ACS)

When ant k is at vertex i, city j is chosen according to the so called *pseudorandom proportional* rule

$$j = \begin{cases} \mathit{argmax}_{l \in \mathcal{N}_i^k} \{\tau_{i,l} [\eta]^\beta\} & \mathit{if}\ q \leq q_o \text{ Exploitation} \\ J & \mathit{otherwise} \text{ Exploration} \end{cases}$$

where q is a random variable uniformly distributed in $[0, 1]$, q_o $(0 \leq q_o \leq 1)$ is a parameter, and J is a random variable selected according to the probability distribution

$$p_{i,j}^k(t) = \begin{cases} \frac{[\tau_{i,j}(t)]^\alpha \cdot [\eta_{i,j}(t)]^\beta}{\sum_{l \in \mathcal{N}_i^k} [\tau_{i,l}(t)]^\alpha \cdot [\eta_{i,l}(t)]^\beta} & \mathit{if}\ j \in \mathcal{N}_i^k \\ 0 & \mathit{otherwise} \end{cases}$$

Local Trail Update

Similar to evaporation

$$\tau_{i,j}(t) \leftarrow (1-\rho)\tau_{i,j}(t) + \triangle\tau_{i,j}(t)$$

but we let $\triangle\tau_{i,j} = (nL_{nn})^{-1}$;

where L_{nn} is the length of a nearest-neighbour tour.

Consequence of Local Trail Update

- Desirability of edges changes dynamically
- Edge pheromone level decreases
- As edges are visited by different ants the desirability to visit lessens
- Exploration of unvisited edges becomes more favourable and in turn ants visit vertices at different times in the tour (for example, ant m_1 will not visit the early visited vertices of ant m_2's tour.)

Global Trail Update

After each iteration the best ant is chosen to update it's tour by setting additional pheromone inversely proportional to the length of it's tour

$$\tau_{i,j}(t) \leftarrow (1-\rho)\tau_{i,j}(t) + \rho\frac{1}{L^{ba}} \qquad \forall (i,j) \in T^{ba}$$

where L^{ba} is the length of the best ant's tour and T^{ba} is the tour of the best ant.

AS vs. ACS

- AS deposits pheromone only after completing a tour
- In ACS only the ant with the best tour is allowed to deposit pheromone (globally)
- AS deposits pheromone to all edges
- In ACS the ant with the best tour deposits pheromone only on it's associated tour

Generalized ACO

- Stochastic construction procedure
- Probabilistically build a solution
- Iteratively add solution components to partial solutions via
  - Heuristic information
  - Pheromone trial
- Modify the problem representation at each iteration
- Ants work together and independently
- Leads to good solutions (maybe not optimal but they're good and fast)

```
def ACO_Meta_heuristic():
    while (termination_criterion_not_satisfied):
        schedule_activities:
            ants_generation_and_activity()
            pheromone_evaporation()
            daemon_actions() #optional

def ants_generation_and_activity():
    while (available_resources):
        schedule_the_creation_of_a_new_ant()
        new_active_ant()
```

```
def new_active_ant():
    initialize_ant()
    M = update_ant_memory() #ant lifecycle
    while (current_state != target state)
        A = read_local_ant-routing_table()
        P = compute_transition_probabilities \
                (A, M, problem_constraints)
        next_state = apply_ant_decision_policy \
                (P, problem_constraints)
        move_to_next_state(next_state)
        if (online_step-by-step_pheromone_update)
            deposit_pheromoneon_the_visisted_arc()
            update_ant-routing_table()
        M = update_internal_state()
    if (online_delayed_pheromone_update)
        evaluate_solution()
        deposit_pheromone_on_all_visisted_arcs()
        update_ant-routing_table()
    die()
```

Applications of ACO Algorithms

Problems Solvable with ACO

- Vehicle Routing Problem (and variations)
- Assignment Problems
- Scheduling
- Subset problems

Applications of ACO Algorithms

Problems Solvable with ACO

- Vehicle Routing Problem (and variations)
- Assignment Problems
- Scheduling
- Subset problems
* Machine Learning
* Data Network Routing

Applications of ACO Algorithms

Problems Solvable with ACO

- Vehicle Routing Problem (and variations)
- Assignment Problems
- Scheduling
- Subset problems
* Machine Learning
* Data Network Routing

**We see that ACO is especially helpful with dynamic problems*

Advantages and Disadvantages

Advantages

- Good solutions in practical time
- Works well with dynamic and "ill-structured" problems (e.g., network routing problem)

Advantages and Disadvantages

Advantages

- Good solutions in practical time
- Works well with dynamic and "ill-structured" problems (e.g., network routing problem)

Disadvantage

- Optimal solution may not be found
- Difficult theory
- Based off of experimentation

Dynamic Advantage

- The main advantage over other heuristic algorithms such as *simulated annealing* and *genetic algorithm* is that the ant colony algorithm can be run continuously and adapt to changes in real time
- This is especially helpful when the graph may change dynamically such as in *network routing*

Data Network Routing - AntNet

The main ideas of ACO are applied but in an adaptive way.

High Level Description of AntNet Algorithm:[4]

- In certain time intervals *forward-ants* are launched towards random destination vertices
- Ants act independently but communicate indirectly by reading and writing locally to vertices visited
- Ants try to find the minimum cost path from its source to destination
- Ants move in accordance to a stochastic policy based on
  1. local ant generated and maintained information
  2. local problem dependent heuristic information
  3. ant private information
- When arrived at destination, *backward-ants* are launched back to the source along the same path but in opposite direction
- While backtracking, network status and routing table of each visited vertex is modified as a function of the path taken and of its goodness
- Once returned, the ant is destroyed and new ants are born.

[4] AntNet: Distributed Stigmergetic Control for Communications Networks

Things To Consider When Implementing ACO Algorithms

- How many ants to use?
- How to balance ant exploration vs exploiting the pheromone trails?
- Heuristic choices?
- When to update pheromones?
- Which ants should update pheromones?
- Stopping criteria?

Final Conclusions

- (summary)

References

G. Di Caro, M. Dorigo (1998)

Ant Net: Distributed Stigmergetic Control for Communications Networks

Journal of Artificial Intelligence Research 9(1998), 317 – 365.

M. Dorigo, G. Di Caro, L. Gambardella (1999)

Ant Algorithms for Discrete Optimization

Artificial Life

M. Dorigo, A. Colorni (1996)

The Ant System: Optimization by a colony of cooperating agents

IEEE Transactions on Systems, Man, and Cybernetics-Part B Vol.26, No.1, 1 – 13.

M. Dorigo, T. Stutzle (2004)

Ant Colony Optimization

MIT Press

A.E. Rizzoli, F. Oliverio, R. Montemanni, L.M. Gambardella (2012)

Ant Colony Optimisation for vehicle routing problems: from theory to applications

Publisher Name ??(?), xx – yy.

References

C. Blum (2005)

Ant colony optimization: Introduction and recent trends

Physics of Life Reviews 2, 353 – 373.

M. Juenger, G. Reinelt and G. Rinaldi (2003)

Exact algorithms for NP-hard problems: A survey

Combinatorial Optimization - Eureka! You shrink! 185 – 207

The End