Ant Colony Optimization

ACO

- developed by Dorigo
- ant algorithms
  - study models derived from observations of real ants
  - use models for developing algorithms to solve optimization problems
- ACO targets discrete optimization problems
- a population-based SLS method

ACO

- ants: simple agents with basic properties
- each one of k ants handles a candidate solution
- ants coordinate their activities through indirect communication mediated by the modification of the environment in which they move (stigmergy)

ACO

- ants find shortest path from food to nest using pheromone trails
  - isolated ant moves randomly
  - ant follows pheromone trails $\Rightarrow$ reinforces trail
- probability of using a trail increases as more ants choose it (due to the pheromone deposited by the ants)
- pheromones evaporate with time

ACO

- autocatalytic behavior emerges
  - as more ants follow trail, it becomes more attractive
  - a positive feedback
  - a process that reinforces itself causing rapid convergence

ACO

- artificial ants:
  - have memory
  - not completely blind
  - time is discrete
- Simple ACO: S-ACO
- ACO is a construction heuristic

S-ACO

- each ant builds a solution from source to destination
- at each step, a decision policy is used
- decisions based on local information at each node
- decisions made stochastically
- ants communicate through stigmergy

Problem Representation

a minimization problem (S,f,Ω)

S : set of candidate solutions

F : objective function (cost)

Ω : set of constraints

s* : globally optimal, feasible solution with minimum cost

Problem Representation

(S,f,Ω) is mapped onto a problem with following characteristics:

- $C=\{c_1, c_2, \ldots, c_{N_C}\}$: finite set of components
- $\mathcal{X}$: set of all possible states
- $\chi=\{c_i \ c_j \ \ldots \ c_h\}$: state of the problem given as sequences of elements of C
- feasible / infeasible states
- g(s) : cost of a candidate solution $s \subseteq S$

Problem Representation

$G_c=(C,L)$: construction graph

- nodes are components (C)
- connections are (L)
  - L fully connects the graph

Problem Representation

- ants construct solutions through randomized walks on G_c=(C,L)
- Ω (constraints) implemented through decision policies of ants
  - sometimes ants are only allowed to construct feasible solutions

Problem Representation

- components and connections may have an associated
  - pheromone trail : τ_i (if associated to components)
    τ_{ij} (if associated to connections)
  - heuristic value : η_i / η_{ij}
- pheromone trails provide long-term memory about the whole ant search process
- pheromone trails updated by ants
- heuristic value represents a priori information about the problem instance definition
  - usually cost of adding a component / connection

Ant's Behavior

- each ant k of the colony has these properties:
  - exploits construction graph to search for optimal solutions
  - has memory M^k where it stores info on path followed so far which is used for:
    - building feasible solutions
    - computing η
    - evaluating the solution found
    - retracing the path backwards to deposit pheromone

Ant's Behavior (cont.)

- each ant k of the colony has these properties:
  - has a starting state (usually an empty set or a single component sequence) and one or more termination criteria
  - when in a state x_r, it moves to a node in its neighborhood
  - stops when a termination criterion is satisfied
  - usually infeasible solutions are not permitted

Ant’s Behavior (cont.)

- it selects the next move using a probabilistic decision rule based on
  - locally available pheromone trails and heuristic values
  - ant’s private memory storing its current state (past history)
  - problem constraints
- when it adds a solution component / connection, it can update the associated pheromone trail
- when solution construction is completed, it retraces its steps and updates all pheromone trails along its path

Ant’s Behavior (cont.)

It is important to note:

- Ants move concurrently and independently.
- Each ant is complex enough to find a (probably poor) solution to the problem under consideration.
- Typically, good quality solutions emerge as the result of the collective interaction among the ants which is obtained via indirect communication mediated by the information ants read/write in the variables storing pheromone trail values.

ACO Algorithm

- has 3 procedures
  - ConstructAntSolutions: manages colony of ants moving on $G_c=(C,L)$
  - UpdatePheromones: modifies pheromone trails (add pheromone / forget through evaporation of pheromones)
  - DaemonActions: implements centralized actions which cannot be performed by single ants, such as
    - activation of LocalSearch (optional procedure) procedure
    - deciding if some trails need extra deposit of pheromones
    - determining which ants should deposit extra pheromones, ...

ACO Outline

```
procedure ACO(p')
  input: problem instance p'∈P
  output: solution s'∈S'(p') or ∅

  sp:={∅}; //population of k ants
  s':= ∅;
  f(s'):=∞;
  τ:=initTrails(p');
  while not terminate(p',sp) do
    sp:=construct(p',τ,η);
    sp':=localSearch(p',sp); //optional
    if (f(best(p',sp'))<f(s') then
      s':=best(p',sp');
    end
    τ:=updateTrails(p',sp',τ);
  end
  if (s' ∈S')then
    return s';
  else
    return ∅;
  end
end ACO.
```

Note: Good parameter settings found in literature!

Applications of ACO

- TSP
- vehicle routing
- sequential ordering
- quadratic assignment
- graph coloring
- generalized assignment
- university course timetabling
- job/open/flow shop
- project scheduling
- bin packing
- fuzzy systems
- classification rules
- total tardiness
- total weighted tardiness
- multidimensional knapsack
- maximum independent set
- redundancy allocation
- set covering
- maximum clique
- shortest common supersequence
- constraint satisfaction
- protein folding
- network routing
- ...

How to Apply ACO

- Traveling Salesman Problem: TSP (√)
- Generalized Assignment Problem: GAP (√)
- Multidimensional Knapsack Problem: MKP (√)

ACO for the TSP

- TSP: finding minimum length Hamiltonian cycle of graph
- TSP is the application chosen when the first ACO algorithm, Ant System (AS), was proposed
- G=(N,A) : problem graph
  - N: n cities
  - A: arcs fully connecting nodes;
  - d_{ij}: weights of arcs (distances)
- solution: permutation of cities

- pheromone trails and heuristic info:
  - τ_{ij}: desirability of visiting city j after i
  - η_{ij}: 1/ d_{ij} (usually)

- solution construction:
  - initially each ant is put on a randomly selected city
  - each ant adds an unvisited node at each step
  - construction terminates when all cities have been visited

- n cities
- $b_i(t)$: number of ants in town i at time t
- m: total number of ants
- ant:
  - chooses next town based on distance and pheromone trail
  - has a tabu list (list of visited towns)
  - lays pheromone trail when tour is completed

- $\tau_{ij}(t)$: intensity of trail on edge (i,j) at time t
- iteration: m moves during interval (t, t+1) by m ants
- each ant completes tour after n iterations
- when tour is completed, trail intensities updated

$$\tau_{ij}(t+n) = \rho * \tau_{ij}(t) + \Delta\tau_{ij}$$

$$\Delta\tau_{ij} = \sum_{k=1}^{m} \Delta\tau_{ij}^{k}$$

where

- ρ: coefficient such that (1- ρ) represents evaporation of trail between time t and t+n (must be <1 to avoid unlimited accumulation of pheromones)
- $\Delta\tau_{ij}^{k}$: quantity per unit of pheromone laid on edge (i,j) by ant k between time t and t+n

$$\Delta\tau_{ij}^{k} = \begin{cases} \dfrac{Q}{L_k} & \text{if kth ant uses edge} \\ 0 & \text{otherwise} \end{cases}$$

where

- Q is a constant
- L_k is tour length of ant k

transition probability for ant k from town i to town j:

$$p_{ij}^{k}(t) = \begin{cases} \dfrac{[\tau_{ij}(t)]^{\alpha} * [\eta_{ij}]^{\beta}}{\sum_{s \in allowed_k} [\tau_{ij}(t)]^{\alpha} * [\eta_{ij}]^{\beta}} & \text{if } j \in \text{allowed}_k \\ 0 & \text{otherwise} \end{cases}$$

where

- η_{ij}: visibility=$1/d_{ij}$
- $allowed_k$={N-$tabu_k$} town collection ant k can select
  - $tabu_k$ is the taboo list of ant k, indicating the town collection where ants are not selectable
  - N is the total number of towns from the current location to the next location
  - $tabu_k$ is constantly changing as the position of the ant changes.
- α and β control relative importance of trail versus visibility, respectively

transition probability is a trade-off between choosing shortest path and most traveled path

Pseudocode of ACO for TSP

```
1) Initialize
     set t=0   (time counter)
     set NC=0  (cycles counter)
     for all edges (i,j)
        set τij(0)=c and Δτij=0
     place m ants on n nodes

2) set s=1  (tabu list index)
    for k=1 to m do
        place starting town of ant k in tabuk(s)

3) repeat until tabu list full    (repeated n-1 times)
        set s=s+1
        for k=1 to m do
            choose town j with probability pij^k(t)
            move ant k to town j
            insert town j in tabuk(s)
```

```
4) for k=1 to m do
        move ant k from tabuk(n) to tabuk(1)
        compute length of tour for ant k (Lk)
        update shortest tour found
        for every edge (i,j)
          for k-1 to m do
            calculate Δτij^k
            Δτij = Δτij + Δτij^k

5) for every edge (i,j)
        compute τij(t+n)=ρ* τij(t)+ Δτij
    set t=t+n
    set NC=NC+1
    for every edge (i,j)
        set Δτij=0

6) if (NC < NCmax)and (not stagnation behavior) then
        empty all tabu lists
        go to step 2
    else
        print shortest tour
            stop
```

ACO for the Knapsack Problem

- construction graph
  - C: set of items
  - L: fully connects the set of items
  - profit of adding an item may be assumed with components or connections
- constraints
  - resource constraint may be handled during solution construction (i.e., not allow inclusion of items violating the resource constraint)

- pheromone trail update
  - τ_i associated with the components: gives desirability of adding item i to current partial solution
- heuristic information
  - heuristic information should prefer items with high profits and low resource requirements

- solution construction
  - each ant adds items based on τ_i and η_i probabilistically to its path
    - each item may be added only once
  - construction ends when an ant cannot add more items without violating any constraints
  - this means that each ant may have solutions of different lengths !