Nature-Inspired Computing

Ant Colony Optimization

Dr. Şima Uyar
September 2006

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 ⇒ 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
- double-bridge experiments

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

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

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
- 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 $s \subseteq S$

Representation

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

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

ACO

- 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

ACO

- components and connections may have an associated
  - pheromone trail : τ_i / τ_{ij}
  - heuristic value : η_i / η_{ij}
- pheromone trails provide long-term memory about whole of the ant search
- pheromone trails updated by ants
- heuristic value is an apriori info
  - usually cost of adding a component / connection

ACO

- 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 found solution
    - retracing the path backwards

ACO

- 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, moves to a node in its neighborhood
- stops when a termination criterion is satisfied
- usually infeasible solutions are not permitted

ACO

- ants select next move using a probabilistic decision rule based on
  - locally available pheromone trails and heuristic values
  - ant's private memory storing its current state
  - 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

ACO

! Ants act concurrently and independently. Each ant finds usually a poor quality solution to the problem. Through indirect communication between ants, good quality solutions emerge. **!**

ACO

- 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 done by single ants, such as activation of LocalSearch (optional procedure) procedure or decide if some trails need extra deposit of pheromones, or determine 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 time-tabling
- job/open/flow shop
- project scheduling
- bin-packing
- fuzzy systems
- classification rules
- total tardiness
- total weighted tardiness
- multi-dimensional knapsack
- maximum independent set
- redundancy allocation
- set covering
- maximum clique
- shortest common super-sequence
- constraint satisfaction
- protein folding
- network routing
- ...

How to Apply ACO

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

ACO for the TSP

- TSP: finding minimum length Hamiltonian circuit 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}: weight 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
- $b_i(t)$: no of ants in town i at time t
- m: total no 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

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

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

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

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

where
- η_{ij}: visibility=1/d_{ij}
- allowed_k={N-tabu_k}
- α and β control relative importance of trail versus visibility

transition probability is a trade off between choosing shortest path and most travelled 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 GAP

Problem Definition: a set of tasks $i \in I$ have to be assigned to a set of agents $j \in J$. Each agent has a limited capacity a_j and each task i assigned to agent j consumes r_{ij} amount of agent's capacity. Cost of assigning task i to agent j is d_{ij}.

The objective is to find a feasible assignment of tasks with minimum cost.

- in the construction graph
  - set of components = set of tasks and agents, $C=I \cup J$
  - each assignment, consisting of n couplings of (i,j) tasks and agents corresponds to an ant's walk

- constraints
  - ant walks alternatingly from a task node to agent nodes without repeating a task node (agent nodes can be repeated)
  - resource capacity constraints enforced through appropriately defined neighborhoods (allow only feasible movements)

- pheromone trails and heuristic information
  - during solution construction ants make two decisions:
    - choose task to assign next
    - choose agent to assign task to
  - pheromone trail can be associated with both:
    - learn the order of task assignments
    - learn the desirability of assigning a task to an agent
  - pheromone trail can be associated with both:
    - e.g. bias task assignment towards those that use more resources
    - e.g. bias choice of agents with smaller costs and smaller resource use

- solution construction
  - choose component based on τ_{ij} and η_{ij} and the capacity constraints

ACO for the MKP

- 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 constraints may be handled during solution construction (i.e. not allow inclusion of items violating any resource constraints)

- 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

---

when determining heuristic information

- possible to calculate average resource requirements

  $r_{avg}= 1/m * \Sigma r_{ij}$ for each item and then set $\eta_i=b_i/ r_{avg}$
- however this ignores tightness of individual resource constraints
- better to include a_j too

  $r_{avg}'= 1/m * \Sigma(a_j/r_{ij})$ for each item and then set $\eta_i'=b_i/ r_{avg}'$

- 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 !

Ant System and Direct Variants

- Ant System (AS)
- Elitist Ant System (EAS)
- Ran-Based Ant System (AS_{rank})
- Max-Min Ant System (μμAS)

EAS

- aim is to provide reinforcement to arcs belonging to best tour found since the beginning
  - more pheromone deposited for the best-so-far tour
  - a daemon action

AS_{rank}

- each ant deposits pheromones in amounts decreasing based on its rank
- also as in EAS, best-so-far tour deposits most in each iteration

μμAS

4 modifications to AS

- stongly exploits best tours found: only the iteration-best ant or the best-so-far ant is allowed to deposit
  - has effect of limiting probability of selecting a city j after city i
  - may lead to stagnation, i.e. all ants follow same sub-ooptimal tour

μμAS

- possible range of τ is limited to [τ_{min}, τ_{max}] to counteract effect of first modification
- τ is initialized to upper limit + low evaporation rate increases exploration at start of search
- τ is re-initialized each time the system approaches stagnation or no improved tours for a consecutive number of iterations

μμAS

- when depositing pheromones
  - only best of iteration
  - only best-so-far
  - or both (each with a given frequency)
- experimets show that
  - for small sized TSP problems, iteration-best works best
  - for larger instances alternating between iteration-best and best-so-far works best

Extensions of Ant System

- Ant Colony System (ACS)
- Approximate Non-Deterministic Tree Search (ANTS)

ACS

differs from AS in 3 points

- exploits the search experience of ants more through the action choice rule
- pheromone evaporation and pheromone deposit only on arcs belonging to best-so-far tour
- each time an ant uses an arc to move from i to j, it removes some pheromones from the arc to increase exploration of different paths

ANTS

- computes lower bounds on completion of a partial solution to define the heuristic info
  - when adding an item, compute lower bound on time for completion of solution if item is added ⇒ heuristic info
  - helps add items which otherwise will be ignored if they currently have high costs

ANTS

- has a novel action selection rule
- has a modified pheromone trail update method

Comparison

- if solution quality is more important ⇒ use μμAS
- if achieving acceptable solutions faster is more important ⇒ use ACS