Ant Colony Optimization

Part 3: The ACO Metaheuristic

Fall 2009

Instructor: Dr. Masoud Yaghini

Outline

- Introduction
- Problem Representation
- Ants' Behavior
- The Metaheuristic
- The Traveling Salesman Problem
- The Generalized Assignment Problem
- The Multiple Knapsack Problem
- References

Introduction

ACO Metaheuristic

- **Ant colony optimization is a metaheuristic**
  - in which a colony of artificial ants cooperate in finding good solutions to difficult discrete optimization problems.
- Artificial ants communicate indirectly by stigmergy, that is, by indirect communication mediated by the environment
- ACO algorithms can be used to solve both **static** and **dynamic** combinatorial optimization problems.

ACO Metaheuristic

- **Static problems**
  - are those in which the characteristics of the problem are given once and for all when the problem is defined, and do not change while the problem is being solved.
  - An example of such problems is the TSP, in which city locations and their relative distances are part of the problem definition and do not change at run time.

ACO Metaheuristic

- **Dynamic problems**
  - are defined as a function of some quantities whose value is set by the dynamics of an underlying system.
  - The problem instance changes therefore at run time and the optimization algorithm must be capable of adapting online to the changing environment.
  - An example of this situation is **network routing problems** in which the data traffic and the network topology can vary in time.

Problem Representation

Problem Representation

- An artificial ant in ACO is a stochastic constructive procedure that incrementally builds a solution by adding opportunely defined solution components to a partial solution under construction.
- The ACO metaheuristic can be applied to **any** combinatorial optimization problems
- The real issue is **how to map** the considered problem to a representation that can be used by the artificial ants to build solutions.

Problem Representation

(S, f, Ω)	The minimization problem
S	The set of candidate solutions
f	The objective function (cost)
Ω	The set of constraints
$f(s, t)$	An objective function (cost) value to each candidate solution $s \in S$, and $\Omega(t)$ is a set of constraints at time t.
s	Index for candidate solution, $s \in S$
s^*	A globally optimal feasible solution, that is, a minimum cost feasible solution to the minimization problem.
C	A set of components, where $C=\{c_1, c_2, \ldots, c_{Nc}\}$, and Nc is the number of components.

Problem Representation

x	The **state** of the problem is defined in terms of sequences of elements of C, $x = <c_i, c_j, \ldots, c_h, \ldots>$
$\|x\|$	The length of a sequence x, that is, the number of components in the sequence
X	The set of all possible states
$\tilde{X}$	A set of feasible states, with $\tilde{X} \subseteq X$
$\tilde{S}$	The set of feasible candidate solutions, with $\tilde{S} \subseteq S$, obtained from S via the constraints $\Omega(t)$.
S^*	A set of optimal solutions, with $S^* \subseteq \tilde{X}$ and $S^* \subseteq S$.
$g(s, t)$	A cost is associated with each candidate solution $s \in S$, In most cases $g(s, t) = f(s, t)$

Problem Representation

$G_C = (C, L)$	The **construction graph**, which is completely connected graph, whose nodes are the **components** C, and the **connections** L fully connects the components C.
c_i	The component of i, with $c_i \in C$
l_{ij}	The connection of i and j, with $l_{ij} \in L$
τ	A pheromone trail, τ_i if associated with components, τ_{ij} if associated with connections
η	A heuristic value, η_i if associated with components, η_{ij} if associated with connections. In many cases η is the cost, or an estimate of the cost, of adding the component or connection to the solution under construction.

Implementing the constraints

- The problem constraints $\Omega(t)$ are implemented in the policy followed by the artificial ants, as explained in the next section.
- Implementing the constraints:
  - In a hard way, the ants can build only feasible solutions
  - In a soft way, the ants can build infeasible solutions that can be penalized as a function of their degree of infeasibility.

The Pheromone Trail

- The pheromone trail store in a long-term memory about the entire ant search process, and is updated by the ants themselves.

Ants' Behavior

Ants' Behavior

- Each ant exploits the construction graph $G_C = (C, L)$ to search for optimal solutions $s^* \in S^*$.
- Each ant has a memory M^k that it can use to store information about the path it followed so far.
- Memory can be used to:
  1. Build feasible solutions (i.e., implement constraints Ω)
  2. Compute the heuristic values η
  3. Evaluate the solution found
  4. Retrace the path backward

Ants' Behavior

- Each ant has a **start state** x_s^k and one or more termination conditions e^k.
- Usually, the start state is expressed either as an empty sequence or a single component.

Ants' Behavior

- When in state $x_r = < x_{r-1}, i>$, if no termination condition is satisfied, it moves to a node j in its neighborhood $N^k(x_r)$, that is, to a state $< x_{r-1}, i> \in X$.
- If at least one of the termination conditions e^k is satisfied, then the ant stops.
- When an ant builds a candidate solution, moves to infeasible states are forbidden in most applications

Ants' Behavior

- Each ant selects a move by applying a probabilistic decision rule.
- The probabilistic decision rule is a function of:
  - the locally available pheromone trails and heuristic values
  - the problem constraints.

Ants' Behavior

- When adding a component c_j to the current state, it can update the pheromone trail τ associated with it or with the corresponding connection.

- Once it has built a solution, it can retrace the same path backward and update the pheromone trails of the used components.

The Metaheuristic

The ACO metaheuristic in pseudo-code

```
procedure ACOMetaheuristic
    ScheduleActivities
        ConstructAntsSolutions
        UpdatePheromones
        DaemonActions          % optional
    end-ScheduleActivities
end-procedure
```

The Metaheuristic

- An ACO algorithm can be imagined as the interplay of three procedures:
  - ConstructAntsSolutions
  - UpdatePheromones
  - DaemonActions

ConstructAntsSolutions

- manages a colony of ants that visit **adjacent states** of the considered problem by moving through neighbor nodes of the problem's construction graph G_C.
- They move by applying a stochastic local decision policy that makes use of pheromone trails and heuristic information.
- Ants incrementally build solutions to the optimization problem.

ConstructAntsSolutions

- Once an ant has built a solution, or while the solution is being built, the ant evaluates the (partial) solution that will be used by the UpdatePheromones procedure to decide how much pheromone to deposit.

UpdatePheromones

- is the process by which the pheromone trails are modified.
- The pheromone value can:
  - Increase, as ants deposit pheromone on the components or connections they use, or
  - Decrease, due to pheromone evaporation.
- The deposit of new pheromone increases the probability that a good solution will be used again by future ants.

UpdatePheromones

- Pheromone avoids a too rapid convergence of the algorithm toward a suboptimal region, therefore favoring the exploration of new areas of the search space.

DaemonActions

- **DaemonActions procedure** is used to implement centralized actions which cannot be performed by single ants.
- Examples of daemon actions are:
  - the activation of a local optimization procedure, or
  - the collection of global information
- that can be used to decide whether it is useful or not to deposit additional pheromone to bias the search process from a non-local perspective.

DaemonActions

- As a practical example, the daemon can observe the path found by each ant in the colony and select one or a few ants
  - e.g., those that built the best solutions in the algorithm iteration
  - which are then allowed to deposit additional pheromone on the components/connections they used.

The Traveling Salesman Problem

Traveling Salesman Problem

- The traveling salesman problem is the problem faced by:
  - a salesman who, starting from his home town,
  - wants to find a shortest possible trip through a given set of customer cities,
  - visiting each city once
  - finally returning home.

Traveling Salesman Problem

- The TSP can be represented by a complete weighted graph $G = (N, A)$ with:
  - N : the set of n = |N| nodes (cities)
  - A : the set of arcs fully connecting the nodes.
- Each arc $(i, j) \in A$ is assigned a weight d_{ij} which represents the distance between cities i and j.
- The TSP is the problem of finding a minimum length **Hamiltonian circuit** of the graph, where a Hamiltonian circuit is a closed walk (a tour) visiting each node of G exactly once.

Traveling Salesman Problem

- A solution to an instance of the TSP can be represented as a permutation of the city indices
- This permutation is cyclic, that is, the absolute position of a city in a tour is not important at all but only the relative order is important

Construction Graph

- The construction graph $G_C = (C, L)$, where the set L fully connects the components C, is identical to the problem graph, that is $C = N$ and $L = A$
- Each connection has a weight which corresponds to the distance d_{ij} between nodes i and j.
- The states of the problem are the set of all possible tours.

Constraints

- The only constraint in the TSP is that all cities have to be visited and that each city is visited at most once.
- This constraint is enforced if an ant at each construction step chooses the next city only among those it has not visited yet
- The feasible neighborhood N_i^k of an ant k in city i, where k is the ant's identifier, comprises all cities that are still unvisited.

Pheromone trails and heuristic information

- The pheromone trails are associated with arcs and therefore τ_{ij} in the TSP refer to the desirability of visiting city j directly after i.
- The heuristic information η_{ij} is typically inversely proportional to the distance between cities i and j, a straightforward choice being $\eta_{ij} = 1/d_{ij}$.
- In fact, this is also the heuristic information used in most ACO algorithms for the TSP.

Solution construction

- Each ant is initially put on a randomly chosen start city and at each step iteratively adds one still unvisited city to its partial tour.
- The solution construction terminates once all cities have been visited.

Dynamic Traveling Salesman Problem

- The **dynamic traveling salesman problem (DTSP)** is a TSP in which cities can be added or removed at run time.
- The goal is to find as quickly as possible the new shortest tour after each transition.

Dynamic Traveling Salesman Problem

- **Construction graph**
  - The same as for the TSP: $G_C(C, L)$, where C(t) is the set of cities and L = L(t) completely connects G_C.
  - The dependence of C and L on time is due to the dynamic nature of the problem.

The Generalized Assignment Problem

The Generalized Assignment Problem

- In the **generalized assignment problem (GAP)** a set of tasks i **ϵ** I , has to be assigned to a set of agents j **ϵ** J.
- Each agent j has only a limited capacity a_j and each task i assigned to agent j consumes a quantity r_{ij} of the agent's capacity.
- Also, the cost d_{ij} of assigning task i to agent j is given.
- The objective then is to find a feasible task assignment with minimum cost.

The Generalized Assignment Problem

- Let y_{ij} be 1 if task i is assigned to agent j and 0 otherwise. Then the GAP can formally be defined as

$$\min f(y) = \sum_{j=1}^{m} \sum_{i=1}^{n} d_{ij} y_{ij} \qquad (2.1)$$

subject to

$$\sum_{i=1}^{n} r_{ij} y_{ij} \leq a_j, \quad j = 1, \ldots, m, \qquad (2.2)$$

$$\sum_{j=1}^{m} y_{ij} = 1, \quad i = 1, \ldots, n, \qquad (2.3)$$

$$y_{ij} \in \{0, 1\}, \quad i = 1, \ldots, n, \quad j = 1, \ldots, m. \qquad (2.4)$$

The Generalized Assignment Problem

- The constraints in equation (2.2) implement the limited resource capacity of the agents,
- The constraints given by equations (2.3) and (2.4) impose that each task is assigned to exactly one agent and that a task cannot be split among several agents.

Construction graph

- The problem could be represented on the construction graph $G_C = (C, L)$ in which the set of components comprises the set of tasks and agents, that is, $C = I \cup J$.
- Each assignment, which consists of n couplings (i, j) of tasks and agents, corresponds to at least one ant's walk on this graph and costs d_{ij} are associated with all possible couplings (i, j) of tasks and agents.

Constraints

- Walks on the construction graph G_C have to satisfy the constraints given by equations (2.3) and (2.4) to obtain a valid assignment.
- One particular way of generating such an assignment is by an ant's walk which iteratively switches from task nodes (nodes in the set I) to agent nodes (nodes in the set J) without repeating any task node but possibly using an agent node several times (several tasks may be assigned to an agent).

Constraints

- Moreover, the GAP involves resource capacity constraints that can be enforced by an appropriately defined neighborhood.
- For example, for an ant k, N_i^k could be defined as consisting of all those agents to which task i can be assigned without violating the agents' resource capacity.

Constraints

- If no agent meets the task's resource requirement, then the ant is forced to build an infeasible solution
  - In this case N_i^k becomes the set of all agents. Infeasibilities can then be handled,
  - for example, by assigning penalties proportional to the amount of resource violations.

Pheromone trails and heuristic information

- During the construction of a solution, ants repeatedly have to take the following two basic decisions:
  - (1) choose the task to assign next
  - (2) choose the agent the task should be assigned to.
- Pheromone trail information can be associated with any of the two decisions
  - it can be used to learn an appropriate order for task assignments or
  - it can be associated with the desirability of assigning a task to a specific agent.

Pheromone trails and heuristic information

- In the first case, τ_{ij} represents the desirability of assigning task j directly after task i, while in the second case it represents the desirability of assigning agent j to task i.
- Similarly, **heuristic information** can be associated with any of the two decisions.
  - For example, heuristic information could bias task assignment toward those tasks that use **more resources**, and
  - Bias the choice of agents in such a way that **small assignment costs** are incurred and the agent only needs a relatively small amount of its available resource to perform the task.

Solution construction

- **Solution construction**
  - can be performed as usual, by choosing the components to add to the partial solution from among those that, as explained above,
  - satisfy the constraints with a probability biased by the pheromone trails and heuristic information.

The Multiple Knapsack Problem

The Multiple Knapsack Problem

- Given a set of items i ϵ I with associated a vector of resource requirements r_i and a profit b_i
- The **knapsack problem (KP)** is the problem of selecting a subset of items from I in such a way that they fit into a knapsack of limited capacity and maximize the sum of profits of the chosen items.
- The **multiple knapsack problem (MKP)**, also known as multidimensional KP, extends the single KP by considering multiple resource constraints.

The Multiple Knapsack Problem

- Let y_i be a variable associated with item i, which has value 1 if i is added to the knapsack, and 0 otherwise.
- Also, let r_{ij} be the resource requirement of item i with respect to resource constraint j, a_j the capacity of resource j, and m be the number of resource constraints.
- In the MKP, it is typically assumed that all profits b_i and all weights r_{ij} take positive values.

The Multiple Knapsack Problem

- The MKP can be formulated as

$$\max f(y) = \sum_{i=1}^{n} b_i y_i, \qquad (2.5)$$

subject to

$$\sum_{i=1}^{n} r_{ij} y_i \leq a_j, \quad j = 1, \ldots, m, \quad (2.6)$$

$$y_i \in \{0, 1\}, \quad i = 1, \ldots, n. \qquad (2.7)$$

Construction graph

- In the construction graph $G_C = (C, L)$, the set of components C corresponds to the set of items and, as usual, the set of connections L fully connects the set of items.
- The profit of adding items can be associated with either the connections or the components.

Constraints

- The solution construction has to consider the resource constraints given by equation (2.6).
- During the solution construction process, this can be easily done by allowing ants to add only those components that, when added to their current partial solution, do not violate any resource constraint.

Pheromone trails and heuristic information

- The MKP has the particularity that pheromone trails τ_{ij} are associated only with components and refer to the desirability of adding an item i to the current partial solution.
- The heuristic information, intuitively, should prefer items which have a high profit and low resource requirements.

Pheromone trails and heuristic information

- One possible choice for the heuristic information is to calculate the average resource requirement :

$$\overline{r_i} = \sum_{j=1}^{m} r_{ij}/m$$

- for each item and then to define:

$$\eta_i = b_i/\overline{r_i}$$

- Yet this choice has the disadvantage that it does not take into account how tight the single resource constraints are.

The Multiple Knapsack Problem

- Therefore, more information can be provided if the heuristic information is also made a function of the a_j.
- One such possibility is to calculate:

$$\overline{r_i}' = 1/m \cdot \sum_{j=1}^{m} a_j / r_{ij}$$

- and to compute the heuristic information as:

$$\eta_i' = b_i / \overline{r_i}'$$

Solution construction

- Each ant iteratively adds items in a probabilistic way biased by pheromone trails and heuristic information
- Each item can be added at most once.
- An ant's solution construction ends if no item can be added anymore without violating any of the resource constraints.
- This leads to one particularity of the ACO application to the MKP

References

References

- M. Dorigo and T. Stützle. **Ant Colony Optimization**, MIT Press, Cambridge, 2004.

The End