Ant System: Optimization by a Colony of Cooperating Agents

Brett Harrison and Ameya Velingker

Harvard University
Computer Science 266

February 2, 2010
Introduction

- Last week, we looked at ant behavior
  - Ants will lay pheromone after discovering food source
  - Ants use the levels of pheromone to determine which path to take
  - Behavior leads to selection of shortest path to food source
- Today, we will study an optimization algorithm which is motivated by ant behavior
- This algorithm is called *Ant Colony Optimization* (ACO)
Problem - TSP

- Problem setting - Traveling Salesman Problem (TSP)
- Suppose we have \( n \) cities, with path \( p_{ij} \) between every pair of cities \( \{i, j\} \)
- Length of \( p_{ij} \), i.e. distance between \( i \) and \( j \), is \( d_{ij} \).
- Goal - find path \( P = p_{i_1i_2} \cdots p_{i_ni_1} \) which minimizes

\[
d(P) = d_{i_1i_2} + \cdots + d_{i_ki_{k+1}},
\]

where \( (i_1, \ldots, i_n) \) is a permutation of \( (1, \ldots, n) \)

- Goal restated: find shortest tour which visits every city exactly once, with no repeat
Problem - TSP (continued)

- TSP is classic example of $\mathcal{NP}$-hard problem
- Brute-force method (check every path) is $O(n!)$
- Dynamic programming method yields $O(n^2 2^n)$ algorithm, still slow
- Various heuristic approaches (e.g. local search)
- ACO will provide a heuristic algorithm for solving TSP
Ant System - Ant Properties

- ACO uses multiple agents, called “ants”, which collectively search the space of solutions, and share information in order to find good solutions.

- Properties of ant:
  - Has memory: can store which cities it has visited, distance it has traveled, shortest path it has seen, etc.
  - Has sight: if positioned at city $i$, the ant has knowledge of $d_{ij}$ for all $j \neq i$
  - Traverses one path per discrete time unit
Ant System - Characteristics

- Ants choose the next city to visit with a probability that is a function of the distance to the city and the amount of pheromone on the path to that city.
- After completing a tour, the ant can leave a pre-determined amount of pheromone on any path it has visited, and can sense the amount of pheromone present on any adjacent path.
- Ants are forced to make complete tours by storing previously visited locations in a tabu.
Notation

**Definition (Notation)**

- \( t \) - current time
- \( m \) - number of ants
- \( k \) - ant index
- \( \tau_{ij}(t) \) - the amount of pheromone on the path between \( i \) and \( j \) at time \( t \)
- \( \rho \) - constant of evaporation
- \( \alpha \) - pheromone importance constant
- \( \beta \) - city distance importance constant
- \( Q \) - pheromone quantity constant
- \( Tabu_k(t) \) - tabu for ant \( k \) at time \( t \), which contains visited cities up to time \( t \)
ACO Algorithm

1. Initialize $m$ ants uniformly across the $n$ cities, initialize $\tau_{ij}(0) = c > 0$, initialize $tabu_k$ to contain only starting city for ant $k$.

2. At each time step $t \leq n$, ant $k$ at city $i$ chooses to move to city $j$ (where $j \not\in Tabu_k(t)$) with probability

$$p^k_{ij}(t) = \frac{\tau_{ij}(t)^\alpha \cdot d_{ij}^{-\beta}}{\sum_{j' \not\in Tabu_k(t)} \tau_{ij'}(t)^\alpha \cdot d_{ij'}^{-\beta}}.$$
ACO Algorithm (continued)

3 After $t = n$, each ant completes tour. Let $L_k$ denote the length of the tour taken by ant $k$. The amount of pheromone on each path is updated according to:

$$\tau_{ij}(n) = \rho \cdot \tau_{ij}(0) + \sum_{k=1}^{m} \Delta \tau_{ij}^k,$$

$$\Delta \tau_{ij}^k = \begin{cases} Q/L_k & \text{if } k\text{th ant uses edge } (i,j) \text{ in its tour} \\ 0 & \text{otherwise} \end{cases}.$$ 

4 Best tour $L^*$ is updated, time is reset to $t = 0$. Go to Step 2 and loop, until stopping condition (i.e. reach max iterations)
ACO Algorithm (continued)

- Complexity of algorithm is $O(I \cdot n^2 \cdot m)$, where $I$ is the max number of iterations.
- The authors are able to obtain a near-optimal solution to a 30-city TSP with $I = 2500$.
- The authors also present two alternative algorithms in which, for the definition of $\Delta \tau_{ij}^k$, the quantity $Q/L_k$ is replaced with either $Q$ or $Q/d_{ij}$.
- These two alternatives do not perform as well, which is reasonable since only local information is being considered.
Experimental Study

- Optimization of parameters:
  - \( \alpha \): Importance of trail
  - \( \beta \): Importance of visibility
  - \( \rho \): Trail persistence
  - \( Q \): Measure of amount of trail laid

- Test on *Oliver30* problem (30 cities)

- \( NC_{\text{MAX}} = 5000 \)

- Best parameters: \( \alpha = 1, \beta = 5, \rho = 0.5 \)
Combinations of $\alpha$ and $\beta$

- **Possible behavior**
  - Bad solutions and stagnation (high $\alpha$)
  - Bad solutions but no stagnation (low $\alpha$)
  - Good solutions (central $\alpha$, $\beta$)
Other Effects

- Synergistic effects (number of ants $m$ and $\alpha$)
- Distribution of ants at initialization (uniform distribution vs. random distribution)
- Random distribution performs slightly better
- Elitist strategy
  - Choose $e$ elitist ants
  - Add $eQ/L^*$ to each arc of the best tour
Comparison to Heuristics

- TSP-tailored heuristics
- General purpose heuristics
  - Simulated Annealing
  - Tabu Search

**TABLE III**

Performance of the ant-cycle algorithm compared with other approaches. Results are averaged over ten runs, and rounded to the nearest integer.\(^8\)

<table>
<thead>
<tr>
<th>Method</th>
<th>basic(^1)</th>
<th>2-opt</th>
<th>L-K</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ant-cycle</td>
<td>420</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Near Neighbor</td>
<td>587</td>
<td>437</td>
<td>421</td>
</tr>
<tr>
<td>Far Insert</td>
<td>428</td>
<td>421</td>
<td>420</td>
</tr>
<tr>
<td>Near Insert</td>
<td>510</td>
<td>492</td>
<td>420</td>
</tr>
<tr>
<td>Space Filling Curve</td>
<td>464</td>
<td>431</td>
<td>421</td>
</tr>
<tr>
<td>Sweep</td>
<td>486</td>
<td>426</td>
<td>420</td>
</tr>
<tr>
<td>Random</td>
<td>1212</td>
<td>663</td>
<td>421</td>
</tr>
</tbody>
</table>
Asymmetric traveling salesman problem (ATSP) is a TSP with nonsymmetric distances \( d_{ij} \neq d_{ji} \).

ATSP is harder than TSP:
- TSP can be solved on graphs with \( \sim 1000 \) nodes
- ATSP can be solved optimally only on graphs with a few dozen nodes

AS can be applied to ATSP with no modifications and same complexity.
Robustness

- Autocatalytic AS algorithm can be applied to combinatorial problems

- Requirements
  - Suitable graph representation
  - Autocatalytic feedback process
  - Heuristic
  - Constraint satisfaction method (tabu list)

- QAP and JSP problems
Quadratic Assignment Problem (QAP)

- Assign $n$ facilities to $n$ locations
- Distance matrix $D = \{d_{ij}\}$
- Flow matrix $F = \{f_{h,k}\}$
- An assignment permutation $\pi$ has cost $\sum_{i,j=1}^{n} d_{ij} f_{\pi(i)\pi(j)}$
- Find $\pi$ with minimal cost

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Best known</td>
<td>1150</td>
<td>2570</td>
<td>6124</td>
<td>17212548</td>
<td>88900</td>
</tr>
<tr>
<td>Ant System (AS)</td>
<td>1150</td>
<td>2598</td>
<td>6232</td>
<td>18122850</td>
<td>92490</td>
</tr>
<tr>
<td>AS with non deterministic hill climbing</td>
<td>1150</td>
<td>2570</td>
<td>6128</td>
<td>17212548</td>
<td>88900</td>
</tr>
<tr>
<td>Simulated Annealing</td>
<td>1150</td>
<td>2570</td>
<td>6128</td>
<td>17937024</td>
<td>89800</td>
</tr>
<tr>
<td>Tabu Search</td>
<td>1150</td>
<td>2570</td>
<td>6124</td>
<td>17212548</td>
<td>90090</td>
</tr>
<tr>
<td>Genetic Algorithm</td>
<td>1160</td>
<td>2688</td>
<td>6784</td>
<td>17640548</td>
<td>108830</td>
</tr>
<tr>
<td>Evolution Strategy</td>
<td>1168</td>
<td>2654</td>
<td>6308</td>
<td>19600212</td>
<td>97880</td>
</tr>
<tr>
<td>Sampling &amp; Clustering</td>
<td>1150</td>
<td>2570</td>
<td>6154</td>
<td>17212548</td>
<td>88900</td>
</tr>
</tbody>
</table>
Job-Shop Scheduling Problem (JSP)

- Given a set of $M$ machines and $J$ jobs
- Each job is an ordered sequence of operations from a set $O$
- Each operation has to be processed on a certain machine for a certain amount of time
- Problem is to assign the operations to time intervals to minimize total completion time
- AS can be used to solve JSP within 10% of the optimum on test data sets
Conclusion

- Search methodology based on a distributed autocatalytic process
- Process directed by a greedy force
- Interaction of agents allows greedy force to give suggestions toward an optimal solution
- Employ positive feedback as a search optimization tool
- Show how to apply AS to combinatorial optimization problems such as QAP and JSP