Ant Colony Optimization for TSP

Siddhesh Shirsat[+], Ajinkya Potdar[*], Viviktesh Agwan[%]
108805949[+], 108738610[*], 108943735[%]
Department of Computer Science
Stony Brook University
Stony Brook, U.S.
sishirsat@cs.stonybrook.edu[+], apotdar@cs.stonybrook.edu[*], vagwan@cs.stonybrook.edu[%]

I. Aim

A. To explore the efficiency of Ant Colony Optimization based approach for solving TSP – to achieve minimum or near minimum tour lengths on known standard instances of problem

B. To parallelize the implementation of Ant Colony Optimization approach , in order to speed up the solution

II. Problem Description

A. Given:

- N by N all connected symmetric graph
- Distance between all pairs of cities

B. Problem:

- To find a Hamiltonian circuit i.e. a path that travels through all the cities exactly once and joins back to the first city such that distance covered is minimized

III. Solution

A. Naïve Approach

Traverse each path available on the graph and determine the path with the lowest cost.

1) Advantage:

- Guarantees minimum solution

2) Disadvantage:

- As the number of cities increase, the time required to reach the solution increases exponentially – O (n!).
- Even in a multicore environment, the time of execution is reduced by a constant factor only.

B. Ant Colony Optimization

1) Introduction

Ants use a decentralized, population based mechanism to efficiently search for food in nature. This mechanism is capable of finding a popular route in dynamic and varied environments without any external guidance, control or a central coordination. Ants deposit pheromone on the path they traverse. The pheromone deposition on the path can be used to keep track of the popularity of the edges taken by the ants. This is a collective effort technique and the same result cannot be achieved on an individual basis.

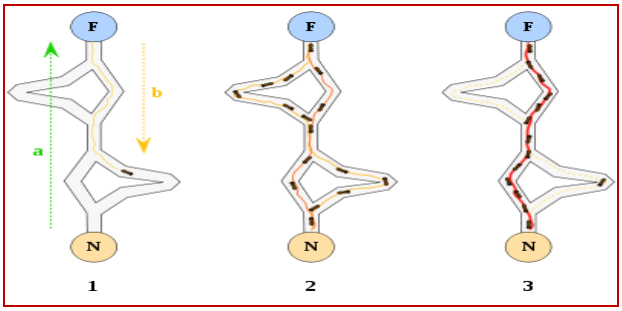

Fig. 1 Ant Colony Optimization

In the figure above, the task is to find the shortest path from N (start) to F (food). Initially, a batch of ants will take random paths thus covering all possible paths. The consequent group of ants will then start taking the lower cost paths as indicated by the amount of pheromone deposited by the earlier ants on the paths.

2) Ant Colony System for TSP

- The algorithm makes use of a set of software agents called artificial ants and their collection in terms of batches.
- The problem is now transformed into the problem of finding the best path on a weighted graph by the artificial ants.
- The construction process is incremental, stochastic and biased by a pheromone model, whose values are modified at run-time by ants.

The pheromone on any given edge is the sum of the pheromone currently remaining (after evaporation) and the pheromone deposited by all the ants who took this path.

The pheromone update is given by:

$$\tau_{i,j} = (1-\rho)\tau_{i,j} + \sum_{k=1}^{m} \Delta\tau_{i,j}^{k} \qquad \forall (i,j) \in L$$

Where

$$\Delta\tau_{i,j} = \begin{cases} \frac{1}{C^k} & \text{if } (i,j) \text{ belongs to } T^k \\ 0 & \text{Otherwise} \end{cases}$$

Algorithm :

a) *Serial ACO for TSP*

1. *For 'a' batches*
   For 'b' ants in each batch

   i. *path p = ant_travel_serial()*
   ii. *Cost c = get cost of path p*
   iii. *Keep track of the minimum cost path encountered for any ant. Minimum cost = min_c, minimum path = min_path*
   iv. *Update the pheromone accumulation for all the edges in this path for every edge e on path p, e→ph_accumulation += (1/c)*

2. *Update pheromone level for all the edges of the graph. For each edge e for the graph*

   e→ph = (1-ρ)edge→ph + e→ph_accumulation
   e→ph_accumulation = 0

3. *Return the minimum cost path*

b) *ant_travel_serial()*

   i. *For every vertex v of the graph, visited[v] = 0*
   ii. *Place the source vertex s in final path p and visited[s] = 1*
   iii. *While all the vertices are not visited*
      - *Find the all the unvisited neighbours of the last visited vertex , calculate probability factor p for each of them as*
      - *p = (pheromone level on the edge)^A * (1/edge weight)^B*
      - *Calculate the sum of all the p's*
      - *Generate a random number r. Select the unvisited neighbour in whose range r falls in as the next visited vertex.*

   iv. *Return the path*

c) *Parallel Ant Colony for TSP*

1. *For 'a' batches*
   ***Parallel_for** 'b' ants in each batch*

   i. *path p = ant_travel_parallel()*
   ii. *Cost c = get cost of path p*
   iii. *Keep track of the minimum cost path encountered for any ant. Minimum cost = min_c, minimum path = min_path*
   iv. *Update the pheromone accumulation for all the edges in this path for every edge e on path p, e→ph_accumulation += (1/c)*

2. *In **parallel** update pheromone level for all the edges of the graph. For each edge e for the graph*

   e→ph = (1-ρ)edge→ph + e→ph_accumulation
   e→ph_accumulation = 0

3. *Return the minimum cost path*

3)Ant Colony System for TSP

Here, every ant selects the minimum cost edge with a certain fixed probability q_0. and goes for the probabilistic determination only with a probability of 1-q_0. This way, we make the algorithm biased towards the selection of minimum cost edge. (Cost depends on both the pheromone deposited so far and the edge weight)

Also, the pheromone model is updated by every ant (local update) and after every batch of ants (global update).

Local Update Rule-

$$\tau_{ij}(t) = (1-\rho)\cdot\tau_{ij}(t) + \rho\cdot\tau_0$$

$$\text{Where } \tau_0 = \frac{1}{(n\cdot L_{nn})}$$

1/(n. L_{nn}) – Distance by Nearest neighbor heuristic

Global Update Rule-

$$\tau_{ij} = (1-\rho)\cdot\tau_{ij} + \rho\cdot\Delta\tau_{ij}$$

$$\text{where } \Delta\tau(r,s) = \begin{cases} (L_{gb})^{-1} & \text{if } (r,s) \in \text{global-best-tour} \\ 0 & \text{otherwise} \end{cases}$$

Algorithm

1. *For 'a' batches*
   Parallel for 'b' ants in each batch

   i. *path p = ant_travel()*

ii. *Cost c = get cost of path p*

Graphs	Size (nodes)	Known Optimal value (C)	Serial ACO
Djibouti	38	6642	6644
Western Sahara	29	27603	28138
Cities of Qatar	194	9352	10497
VLSI XQF131	131	564	590
VLSI XQG237	237	1019	1083
VLSI PKA379	379	1332	1492

iii. *Keep track of the minimum cost path encountered for any ant. Minimum cost = min_c, minimum path = min_path*

iv. *For every edge e on path p,*
$e \rightarrow ph = (1-\rho)\, e \rightarrow ph + (\rho)(1/L_{nn})$

2. *In parallel update pheromone level for all the edges of the graph. For each edge e for the graph*

$e \rightarrow ph = (1-\rho) edge \rightarrow ph + V$

where V = *Cost of best path if* $e \in$ *best path*
$= 0$ *otherwise*

3. *Return the minimum cost path*

Parameters

PARAMETER	SIGNIFICANCE
A	Pheromone factor
B	Edge Weight factor
ρ	Evaporation Constant

Fig. 3 Parameters

1. The pheromone factor:

This parameter controls the extent to which pheromone deposition on the edge will influence the probability of that edge getting selected.

2. Edge Weight Factor:

This parameter controls the extent to which edge weight will influence the probability of that edge getting selected.

3. Pheromone Evaporation factor:

This parameter is used in the local pheromone update and also in global pheromone update. In local pheromone update, its task is to reduce the attraction of most visited routes to prevent convergence. In global pheromone update, its task is to reinforce the best path. Therefore, we select a mediate value, 0.1.

IV. RESULTS

We chose the Geogia tech ACO dataset for benchmarking. This dataset has graphs of different countries. Here node are the cities within country, represented by {x,y} coordinates. Below is the comparison of our results with known optimal values and we can see our results show at the most 1% deviation from optimal results.

Fig. 3 Shortest Tour values

Graphs	Known Optimal value (C)	Parallel ACO	Parallel ACO System
Djibouti	6642	6642	6640
Western Sahara	27603	27797	27586
Qatar	9352	10364	11941
VLSI XQF131	564	590	591
VLSI XQG237	1019	1077	
VLSI PKA379	1332	1437	

Fig. 4 Comparison of shortest tours found by all Algorithms

Graphs	Serial ACO (ms)	Parallel ACO (ms)
Djibouti	76114	62501
Western Sahara	38524	37366
Qatar	2000646	960786
VLSI XQF131	899326	498899
VLSI XQG237	3095783	1394057
VLSI PKA379	more than 1 hour	3274977

Fig. 5 Comparison of running times of serial and parallel Algorithms

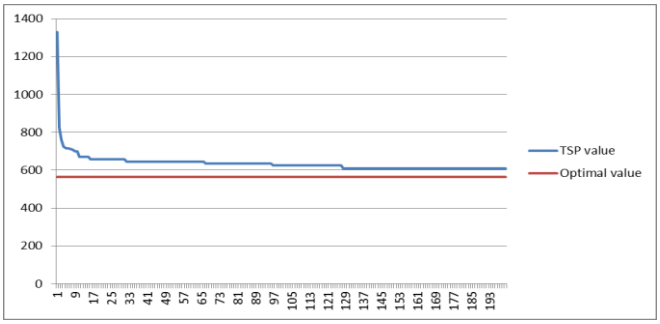

Fig. 6 Convergence of shortest value for VLSI XQF131 (ACO Serial)

V. Applications

A. Shortest Common Super String

- *We can use Ant Colony Optimization algorithm to find the shortest common superstring from a given set of strings.*
- *We treat the substrings as nodes in the graph and the extent of overlap as the cost of the edge between two substrings.*
- *Our aim is to visit all the nodes in the graph keeping the length of the cumulative string minimum.*
- *In our path, from a node we probabilistically choose the neighbor which has maximum overlap. The pheromone level of the path is updated accordingly.*

B. DNA Assembly

- *In shotgun sequencing, whole genomes are sequenced by making clones, breaking them into small pieces, and trying to put the pieces together again based on overlaps.*
- *Fragments are randomly sampled, and thus no positional information is available.*
- *We can use Ant Colony System algorithm for DNA sequence regeneration.*
- *We treat fragments of DNA as nodes in the graph.*
- *Our Aim is to visit all the nodes in the graph.*
- *While visiting nodes we keep track of the superstring constructed so far.*
- *In our path, from a node we probabilistically choose the neighbor which has maximum overlap with the superstring constructed till now. The pheromone level of the path is updated accordingly.*

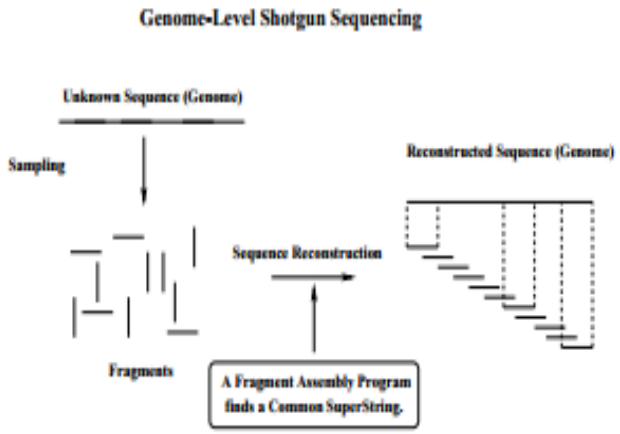

Fig. 7 Shotgun Sequencing

Source: course material of Computational Biology at Stony Brook University by Professor Steven Skiena

Results of DNA Sequencing:

Human DNA sequence
Original DNA sequence length – 3290
Reconstructed Sequence length for a sample size of 500 – 1406
Reconstructed Sequence length for a sample size of 1000 – 2325
Thus, the same amount of information can be constructed in a lesser length.

Chimpanzee DNA Sequence
Original DNA sequence length – 2870
Reconstructed Sequence length for a sample size of 500 – 1235
Reconstructed Sequence length for a sample size of 1000 – 2214
Thus, the same amount of information can be constructed in a lesser length

VI. Accomplishments

1) *Successful implementation of serial Ant System*
2) *Successful implementation of parallel Ant System*
3) *Successful running of the above algorithms on graphs of cities and VLSI – (within single digit % variance from known best solution)*
4) *Application of the ACO algorithm on variants of TSP like finding the shortest common superstring, DNA sequence assembly*

VII. Further Work

1) *Application of ACO to other TSP problems like Vehicle Routing Problem, PacMan*
2) *Determining optimal values of parameters (A, B and E) based on the graph*

VIII. Conclusion

1) *The algorithm builds an incremental solution. As the number of batches of ants increases, the solution becomes more refined and approaches towards the optimal solution.*
2) *The Algorithm is as of now able to give solutions which have a single digit % variation from the best known solution of various problem sets.*
3) *The Algorithm also solves the shortest common superstring problem for randomly generated String fragments and DNA fragments.*

References

[1] Optimization, Learning and Natural Algorithms, PhD thesis,(Dorigo, 1992.)

[2] The Ant System: Optimization by a colony of cooperating agents (Dorigo et al., 1991a; Colorni, Dorigo, &Maniezzo, 1992a; Dorigo, 1992).

[3] Ant colony system: a cooperative learning approach to the TSP (Dorigo M. & L.M. Gambardella (1997))

[4] Test Data from: http://www.tsp.gatech.edu/