Using Local Searches Algorithms with Ant Colony Optimization for the Solution of TSP Problems

Renas Rajab Asaad [1] and Nisreen Luqman Abdulnabi[2]

[1] College of Computer Science and Information Technology, Nawroz University, Duhok, Kurdistan Region - Iraq
[2] College of Economics and Administrative, University of Duhok, Duhok, Kurdistan Region - Iraq

ABSTRACT

Swarm intelligence is a relatively new approach to problem solving that takes inspiration from the social behaviors of insects and other animals. Ants, in particular, have inspired a number of methods and techniques among which the most studied and successful is the general-purpose optimization technique, also known as ant colony optimization, In computer science and operations research, the ant colony optimization algorithm (ACO) is a probabilistic technique for solving computational problems which can be reduced to finding good paths through graphs. Ant Colony Optimization (ACO) algorithm is used to arrive at the best solution for TSP. In this article, the researcher has introduced ways to use a great deluge algorithm with the ACO algorithm to increase the ability of the ACO in finding the best tour (optimal tour). Results are given for different TSP problems by using ACO with great deluge and other local search algorithms.

KEYWORDS : Travels Salesman Problem (TSP), Ant Colony Algorithm (ACO), Great Deluge Algorithm, Optimization, opt-algorithm.

1. INTRODUCTION

Ant Colony Optimization (ACO) uses the behavior of ants for finding optimal paths for TSP problems. Adding Great Deluge algorithm to ACO increase the efficiency of ACO to get a better result in a minimum time. Great Deluge algorithm generates new tour from an old tour by finding neighbor of the cities in the tour by using local search methods, in this article we used 2-Opt algorithm and N-Shift.[1].

2.Travelling Salesman Problem (TSP)

TSP is an NP-hard problem in combinatorial optimization [1]. Given a set of cities in which every city must be visited once only and return to the starting city for completing a tour such that the length of the tour is the shortest among all possible tours [1, 2]. In general there are two different kinds of TSP, the Symmetric TSP (STSP) and the Asymmetric TSP (ATSP).the number of tours in the ATSP is (n-1)!, Whereas it is (n-1)!/2 in STSP for n cities. Formally, the TSP is a complete weighted graph G (N, A) where N is the set of cities which must be visits, and A (i, j) is the set of arcs connecting the cities together [1]. The length between city A_i and A_j can be represented as d_{ij} .Thus the optimal (minimum length) tour to the TSP can be found as shown below.

$$Btour = \left(\sum_{i=1}^{n-1} d_{p(i)\,p(i+1)} \right) + d_{p(n)\,p(1)}$$

Where p is a probability list of cities with minimum distance between city (p_i and p_{i+1}) [2, 3].

2.1 Ant Colony Optimization (ACO) and TSP

The Ant Colony Optimization (ACO) heuristic is an inspiration of the real ant behavior to find the shortest path between the food and ant's nest [1, 2]. As shown in figure 1.

Academic Journal of Nawroz University (AJNU) Volume 7, No 3 (2018).
Received 16 Feb 2018.
Regular research paper : Published 20 June 2018
Corresponding author's e-mail : renas_rekany@yahoo.com

doi : 10.25007/ajnu.v7n3a193

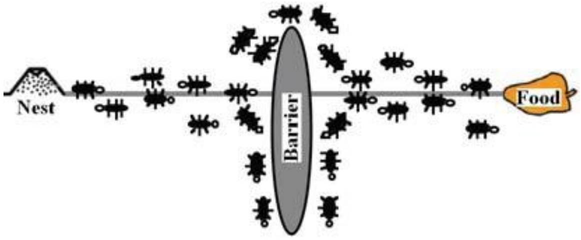

Figure (1) : ACO

The behavior of each ant in nature

- First each ant randomly, laying down a pheromone trail in its path for food searching.
- If any ant founds a food, return to the nest laying down a pheromone trail
- If in a path the pheromone increased the other ant follow that path.

ACO use the same procedure to find the optimal (minimum length) path to the TSP problem, in a given set of cities at first each ant use the pheromone trail to choose a nearest city to its current position and adds cities one by one until it complete the tour by visiting all cities and back to the starting city. After each ant complete its tour ACO update the pheromone trail. ACO pseudo code is shown below.

Initialize
Loop
Each ant is positioned on a starting node
Loop
Each ant applies a state transition rule to incrementally build a solution and a local pheromone updating rule
Until all ants have built a complete solution
A global pheromone updating rule is applied
Until end condition

2.2 Great deluge algorithm

It is a comprehensive approach for solving optimization problems. It use local searches algorithm to find the neighbor of the current solution and compare it with the fitness of the best solution and the water level (WL) if its better it replaces common solution (*New_solution*) with best results (*Best_solution*). This action continues until stop conditions is provided [4]. Great Deluge pseudo code is shown below.

Choose an initial configuration as Old_solution and Best_solution
Choose Δ WL and WL
For n=0 to # of iterations
Generate a New_ solution from neighbor of Old_ solution
If Fitness (New_solution) <WL
If Fitness (New_solution) < (Best_solution)
Old_ solution := New_ solution
End If
End If
WL = WL -ΔWL
End For

2.3 Finding neighbor for Great Deluge Algorithm

In this paper two methods are used to find neighbors to be use by Great Deluge algorithm which are (N-Shift method and 2-Opt method).

2.3.1 N-Shift :

This method changes the order of cities in the current path. It chooses a city and change with all other cities in the list gradually as shown below.

Initial A=> B=> C=> D=>E=>F
Step1 B=>A=> C=> D=>E=>F
Step2 C=>B=> A=> D=>E=>F
⋮

N-Sift pseudo code is shown below
for i=1 to number of cities -1
for j=i+1 to number of cities
replace the position of city(i) and city(j)
End End

2.3.2. 2-Opt algorithm

The 2-opt algorithm basically removes two edges from the tour, and reconnects the two paths in reverse order. This is often referred to as a 2-opt move [5,6,7].

The figure 2 is showing that the tour {1,2,3,4,5,6,7,8} After applying 2-opt algorithm it Become { 1,2,6,5,4,3,7,8,}

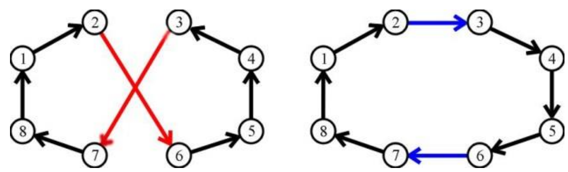

Figure (2) : 2-Opt algorithm

2-Opt pseudo code is shown below.
Require : Tour T .
Let Ti ← Cluster (Ti).
for x ← 1, 2, . . . ,m− 2 do
Calculate the shortest paths along the tour T from every vertex in Ty to every vertex in Tx+1 and from every vertex in Ty+1 to every vertex in Tx for every y = x + 2, x + 3, . . . ,min{m, x + m − 2}.
for y ← x + 2, x + 3, . . . ,min{m, x + m− 2} do
Construct a layered network L as in Figure 2b.
Apply CO to L to get the shortest cycle C.
if w(C) < w(T) then
Replace T with C.
Restart the whole algorithm.

2.4 Using N-Shift and 2-Opt with Great Deluge algorithm

N-Shift or 2-Opt algorithm can be used for (***Generate a New_ solution from the Old_ solution)*** In Great Deluge algorithm. (2-Opt & N-Shift) inside Great Deluge pseudo code is shown below.

Choose an initial configuration as Old_solution and Best_solution
Choose Δ WL and WL

For n=0 to # of iterations
N-Shift OR 2-Opt
If Fitness (New_solution) > WL
If Fitness (New_solution) > (Best_solution)
Old_ solution := New_ solution
End If
End If
WL = WL + ΔWL
End For

2.5 Inserting Great Deluge to ACO

Inserting Great Deluge algorithm to ACO make the result of ACO approaches or equal to the optimal one. Great Deluge with (N-Shift OR 2-Opt) algorithm are used either inside ACO algorithm or at the end of ACO algorithm.

2.5.1 Using Great Deluge inside ACO

When Great Deluge is used inside the ACO algorithm it tries to optimize the results of ACO at each loop. Great Deluge inside ACO pseudo code is shown below.

Initialize
Loop
Each ant is positioned on a starting node
Loop
Each ant applies a state transition rule to incrementally build a solution and a local pheromone ***updating rule***
Apply Great Deluge with (N-shift OR 2-Opt) to the current tour
Until all ants have built a complete solution
A global pheromone updating rule is applied
Until end condition

2.5.2 Using Grete Deluge at the end of ACO

When Great Deluge is used at the end of ACO algorithm it tries to optimize the best solution found by ACO. Great Deluge at the end of ACO pseudo code is shown below.

Initialize
Loop
Each ant is positioned on a starting node
Loop
Each ant applies a state transition rule to incrementally build a solution
and a local pheromone updating rule
Until all ants have built a complete solution
A global pheromone updating rule is applied
Until end condition
Apply Great Deluge with (N-shift OR 2-Opt) to the best tour found by ACO

3. Implementation and Results

This section presents the performance of adapting great deluge algorithm and other local search (N-Shift, 2-Opt) algorithms to the (ACO) algorithm which are thus classified into three different table of results, the first table for the results of ACO algorithm before adding any other algorithm to it and two other tables, a table for the results of Great Deluge and 2-Opt (inside & at the end of) ACO while the other table shows the results of Great Deluge and N-Shift (inside & at the end of) ACO. The results are shown for different TSP problem from (TSPLIB95).

3.1ACO results

Initial ACO results before inserting any other algorithms to it.

Table (1) : ACO results

TSP problem	Optimal Solution	ACO result in 200 iteration	ACO result in 500 iteration
Att48	10628	Fitness 11753	Fitness 11753
ch130	6110	Fitness 6941.6	Fitness 6941.6
berlin52	7542	Fitness 8092	Fitness 8072
ch150	6528	Fitness 6871	Fitness 6871
eil51	426	Fitness 472	Fitness 472
st70	675	Fitness 746	Fitness 746

I.Result of ACO with Great Deluge and N-Shift algorithm

N-Shift is used in two different ways (inside ACO & at the end of ACO) which gives different results. For (att48) the best solution for Great deluge and N-shift inside ACO is (11483) while it was (11904) when great deluge and N-shift are at the end of ACO.

doi : 10.25007/ajnu.v7n3a193

Table (2) : Result of ACO with Great Deluge and N-Shift algorithm

TSP Problem	Optimal Solution	N-shift and great deluge # Iteration=10 inside ACO-TSP # iteration =10	N-shift and great deluge # Iteration=10 at the end of ACO-TSP # iteration =10
Att48	10628	Best tour Fitness 11483	Best tour Fitness 11904
		Elapsed time 179.967442	Elapsed time 5.188908
		Evaluation function 32486881	Evaluation function 11760
ch130	6110	Best tour Fitness 6839	Best tour Fitness 6906
		Evaluation function = 654031301	Evaluation function = 85150
		Elapsed time 4887.513595 seconds	Elapsed time 30.899352 seconds
berlin52	7542	Best tour Fitness 8035	Best tour Fitness 8087
		Elapsed time 235.027542 seconds	Elapsed time 7.219683 seconds
		Evaluation function 41371721	Evaluation function 13780
ch150	6528	Best tour Fitness 6858	Best tour Fitness 6954
		Elapsed time 8025.759414 seconds	Elapsed time 40.799448seconds
		Evaluation function 1.0058e+009	Evaluation function 113250
eil51	426	Best tour Fitness 444	Best tour Fitness 461
		Elapsed time 221.497929 second	Elapsed time 5.615070 second
		Evaluation function 39015511	Evaluation function 13260
st70	675	Best tour Fitness 700	Best tour Fitness 716
		Elapsed time 607.734886 seconds	Elapsed time 9.443444 seconds
		Evaluation function 101430701	Evaluation function 24850

doi : 10.25007/ajnu.v7n3a193

II.Result of ACO with Great Deluge and 2-Opt algorithm

2-Opt algorithm as N-Shift also is used (inside ACO & at the end of ACO) which gives different results. For (att48) the best solution for Great deluge and 2-Opt inside ACO is (10798) while it was (11154) when great deluge and 2-Opt are at the end of ACO.

Table (3) : Result of ACO with Great Deluge and 2-Opt algorithm

Problem	Optimal Solution	2-Opt and great deluge # Iteration=10 inside ACO-TSP # iteration =10	2-Opt and great deluge # Iteration=10 at the end of ACO-TSP # iteration =10
att48	10628	Best tour Fitness 10798	Best tour Fitness 11154
		Elapsed time 422.227777s	Elapsed time 5.999922 seconds.
		Evaluation function 24141096	Evaluation function 32941
ch130	6110	Best tour Fitness 6347	Best tour Fitness 6629
		Evaluation function = 507978272	Evaluation function = 228404
		Elapsed time 8876.979989 seconds	Elapsed time 31.515636 seconds
berlin52	7542	Best tour Fitness 7717	Best tour Fitness 7884
		Elapsed time 647.569437 seconds	Elapsed time 6.764293 seconds.
		Evaluation function 35654729	Evaluation function 45617
ch150	6528	Best tour Fitness 6622	Best tour Fitness 6752
		Elapsed time 9050.060806 seconds	Elapsed time 47.200332seconds
		Evaluation function 836202248	Evaluation function 366145
eil51	426	Best tour Fitness 433	Best tour Fitness 460
		Elapsed time 555.017500 seconds	Elapsed time 6.966547 seconds
		Evaluation function 29675501	Evaluation function 38054
st70	675	Best tour Fitness 696	Best tour Fitness 742
		Elapsed time 1433.249165 seconds	Elapsed time 9.354369 seconds
		Evaluation function 75327200	Evaluation function 63593

doi : 10.25007/ajnu.v7n3a193

6.Conclusions

From the results, its noted that when Great Deluge and 2-Opt are used inside ACO algorithm best tour is almost close to the optimal solution and its better than the other methods. But it need more time and evaluation functions than the other methods.

In general, using Great Deluge and 2-Opt with ACO is much more efficient than using Great Deluge and N-Shift with ACO.

References

1. Almufti, Saman Mohammed. (2015), "U-Turning Ant Colony Algorithm powered by Great Deluge Algorithm for the solution of TSP Problem.".
2. Marco Dorigo, Thomas Stu¨ tzle", (2004), Ant Colony Optimization
3. Federico Greco, (2008), Travelling Salesman Problem
4. J. Basic. Appl. Sci. Res., 2(3)2336-2341, (2012), A New Hybrid Algorithm for Optimization Using PSO and GDA
5. D. Karapetyan, G. Gutin, (2012), Efficient Local Search Algorithms for Known and New Neighborhoods for the Generalized Traveling Salesman Problem
6. Andrej Kazakov, (2009), Travelling Salesman Problem: Local Search and Divide and Conquer working together
7. Alfonsas misevičius, armantas ostreika, antanas šimaitis, vilius žilevičius, (2007), vol.36, no.2, improving local search for the traveling salesman problem.
8. Almufti, S. Mohammed (2017), " Using Swarm Intelligence for solving NP-Hard Problems", Academic Journal of Nawroz University, doi (https ://doi.org/10.25007/ajnu.v6n3a78).