Traveling Salesman Algorithms Complexity

Fatima Thaher Aburomman
Al-Balqa Applied University,
Salt, Jordan

ABSTRACT

The travelling salesman problem (TSP) is widely studied in computer science. There is a practical importance, and can be applied to solve many practical daily lives problems, so many algorithms developed to solve this problem, each with its efficient. Insertion, genetic, greedy, greedy 2-opts and nearest neighbor, are all algorithms used to solve (TSP). This paper will study these algorithms and present the main differences between these algorithms according to its complexity, and which one is the most efficient to solve the (TSP)

Keywords

TSP complexity using Insertion, TSP using Greedy, TSP using Genetic

1. INTRODUCTION

The traveling salesman (TSP) problem is an important famous optimization problem, appeared in research area in 1920's. [8] The main idea of (TSP) is a traveling salesman wants to visit each of n cities exactly once and return to his starting city. In which order should he visit these n cities to minimizing the total distance travelled. Since the (TSP) have both practical and theoretical importance, many algorithms have been developed to produce solutions near to the optimal one for (TSP), but we looking for the efficient algorithm that solve this problem and produce the nearest solution to the optimal. This paper will present a detailed comparative study on results of these algorithms that solve (TSP), and which of these algorithms the efficient one is. The rest of this paper is organized as follow: we will explain some of previous works that done to solve (TSP) in order to understand the growth of solutions that achieved; this will be introduced in the related work section. In the experiments section we will introduce some algorithms that solve (TSP), in order to facilitate comparisons and results in the result section. The last section will conclude our studying and analysis on this paper.

2. RELATED WORK

From 1920's until now different works have been done to solve (TSP), Hmaifar introduced the straight forward way to solve the (TSP) which examine all possible tours and evaluating their corresponding tour length. The tour with smallest length is selected as the best, which is guaranteed to be optimal. Hmaifar, said that" one approach would certainly find the optimal solution of any (TSP) is the application of exhaustive enumeration and evaluation". [1] Fischer and Richer (1982) used a branch and bound approach to solve a (TSP) with two (sum) criteria. Gupta and Warburton (1986) used the 2-and 3-opt heuristics for the max ordering (TSP). Sigal (1994) proposed a decomposition approach for solving the (TSP) with respect to the two criteria of the route length and bottlenecking, where both objectives are obtained from the same cost matrix. [3] Tung (1994) used a branch and bound method with a multiple labeling scheme to keep track of possible Pare to optimal tours. Melamed and Sigal (1997) suggested an e-constrained-based algorithm for bi-objective (TSP). Ehrgott (2000) proposed an approximation algorithm with worst case performance bound. Hansen (2000) applied the tabu search algorithm to multi objective (TSP). Borges and Hansen (2002) used the weighted sums program to study the global convexity for multi-objective (TSP). Jaszkiewicz (2002) proposed the genetic local search which combines ideas from evolutionary algorithms, local search with modifications of the aggregation of the objective functions. Paquete and Stuzle (2003) proposed the two-phase local search procedure to tackle bi-objective (TSP). During the first phase, a good solution to one single objective is found by using an effective single objective single objective algorithm. This solution provides the starting point for the second phase, in which a local search algorithm is applied to a sequence of different aggregations of the objectives, where each aggregation converts the bi-objective problem into a single objective one. Yan et al (2003) used an evolutionary algorithm to solve multi objective (TSP). Angel, Bampis and Courves (2004) proposed the dynasearch algorithm which uses local search with an exponential sized neighborhood that can be search in polynomial time using dynamic programming and rounding technique. Paquete, Chiarandini and Stutzle (2004) suggested a Pareto local search method which extends local search algorithm for the single objective (TSP) to bi-objective case. This method uses an archive to hold non-dominated solutions found in the search process.

3. EXPERIEMENTS

The traveling salesman problem can be expressed mathematically as follows:If we have a graph G = (v,e) and the weight Cij on the edge between nodes i and j is a non-negative value. Find a tour of minimal cost. The first method was known to optimally solve the travelling salesman problem of any size, was by enumerating each possible tour and searching for the tour with smallest cost. Each tour has a size of 1 2 3... N, where n is the number of cities, so the number of tours will be n! If N is large, it becomes impossible to find the cost of every tour. If we could identify and evaluate one tour per nanosecond (on one billion tours per second), it would require almost ten million years (number of possible tours = 3.2∗1023) to evaluate all of the tours in a 25-city (TSP).

3.1 (TSP) Algorithms

3.1.1 Insertion Algorithm

All insertion algorithms start with a tour consisting of an n city then in each step chooses a city k not in the tour. This city is inserted between two cities i and j such that the insertion cost d(i,k) + d(k,j) = d(i,j) is minimized. This algorithm stops when all cities are in the tour. Insertion Algorithm Code:

```
begin
min = ∞
 for m = 1, ......n.do
begin
Tm = φ; Cm=m
while Cm 6= N do
begin Step A: let k1 = min (k, (n−Cm));
let Δ(s∗) =min(Δ(s): s⊆ N-Cm and s=k1); Cm=Cm+S*;
```

```
If min> d(Tm) then min=d(Tm);
T: =Tm
end
end
```

Insertion algorithm has a time complexity of O(n2k+2).

3.1.2 Greedy Algorithm

Greedy algorithm is the simplest improvement algorithm. It starts with node 1. Then the algorithm calculates all the distances to other n-1 nodes. Go to the next closest node. Take the current node as the departing node, and select the next nearest node from the rest n-2 nodes. This process continues until all the nodes are visited once and only once then back to node 1. when the algorithm is finished the sequence is returned as the best tour. This algorithm is useful because of its simplicity to implement and understand. When the problem size is small it leads to a good solution. It saves much computational time because it doesn't't make any exchange of nodes. begin

```
T: = φ
while T is not a tour do
begin d(e)=min(d(f): f ∈ n2 and T ∪f is contained in at least one tour T: = T ∪e
end
end
```

Greedy algorithm has a time complexity of θ (log n).

3.1.3 Greedy 2-opt algorithm

The greedy 2-opt algorithm consists of three steps: Step1: Let S be the initial solution provided by the user and Z Its objective function value. Set S*=s, Z*=z, i=1 and j=i+1=2. Step2 Transpose Node i and Node j, i¡j. Compare the result z with Z*.If z¿=Z*,set S*=s, Z*=z, i=1, j=i+1=2 and go to step 2.If z ¿=Z*and j=n, set i=i+1,j=j+1 and repeat step 2.otherwise, output S* as the best solution and terminate the process. Greedy 2-opt algorithm also considers pair wise exchanges. Initially, it transposes nodes 1 and 2. If the result is less than the previous one, two nodes are immediately transposed. Else the algorithm will go on to node 3 and evaluate the exchange, and so on. Instead of adding one node at a time, one added the minimum length set of k nodes at each stage, in this case the time complexity will be O(n).

3.1.4 Genetic Algorithm

Genetic Algorithms consists of the following steps: 1. Choose initial population randomly. 2. Evaluate the fitness of each individual in the population. 3. Repeat until termination. 4. Select individuals to reproduce. 5. Create new generation throw crossover or/and mutation and give off spring. 6. Evaluate the individual fitness of the off spring. 7. Replace worst part of population with off spring. Genetic algorithm has a time complexity of O (log n).

3.1.5 nearest neighbor algorithm

This algorithm is like the greedy. For a simple path P, we consider i(p), j(p) the initial and the terminal nodes of P. We allow a single ton node to be a path with no edges[?].

```
begin
min: =∞
for m=1, ........n do
begin Tm: =φ; i(Tm): =j;(Tm): =m;
while Tm< n-1 do
begin
d(e)= min (d (x, y)) = x is not Tm and y=i(Tm) or y is not Tm and x=j(Tm)
Tm: = Tm+ e
end
Tm: =Tm+(j(Tm), i(Tm));
if d(Tm)<min then T: =Tm; Min: =d(Tm)
end
end.
```

4. RESULTS AND CONCLUSION

In this section we present some results obtained from experiments section. Algorithm complexity identifies the time and space that algorithm needs to accomplish its work. So genetic and greedy algorithm have the best complexity which is O (log n), but when the problem size is very large it is suitable to use greedy algorithm more than using genetic, because genetic will be complex to be applied. Then the greedy 2-opt which need O (n) is better than insertion algorithm that need O(n2k+2).

5. REFERENCES

[1] Abdullah Homaitor Shanyunchauna,and Gunar E.liepins.Scherna analysis of the travelling salesman problem using genetic algorithms,complex system.

[2] Fisher,R,Richter,K..Solving amulti objective Travelling Sales man problem by dynamic programming.

[3] Hansen,M.Puse of substitute Scalarizing functions to guide alocal search based heuristics the case of MOTSP.Journal of Heuristics6(2000)419-431

[4] Yan,Zhang L.kang Anew MOEA for multi-objective TSP and it's convergence property analysis.

[5] Alba, E. Parallel Metaheuristics: A New Class of Algorithms. Wiley Series on Parallel and Distributed Computing. Wiley-Interscience, Hoboken, NJ, 2005.

[6] Gutin G., A. Punnen. The Traveling Salesman Problem and Its Variations, Kluwer Academic Publishers, 2004.

[7] Reeves C., J. Rowe, Genetic Algorithms . Principles and Perspectives: A Guide to GA Theory, Springer, 2002.

[8] G. Bendall and F. Margot, Greedy Type Resistance of Combinatorial Problems, Discrete Optimization 3 (2006), 288298.

[9] P. H. Chen, S. Malkani, C.-M. Peng, and J. Lin. Fixing antenna problem by dynamic diode dropping and jumper insertion. In Proc. of ISQED, 2000.