1 INTRODUCTION

The Traveling Salesman Problem (TSP) is an NP-Complete computational problem which has been studied extensively for several decades [1]. In this paper, techniques for obtaining/approximating the solution to the symmetric Traveling Salesman Problem are developed and compared. A branch-and-bound algorithm using a minimum spanning tree to inform a lower bound on sub-problems is presented; which, given enough time, produces an exact solution to the problem. The use of approximation algorithms proves more prudent for larger problems while sacrificing solution quality; a fact which motivated the development of a greedy local search algorithm which explores 1-exchange neighbors. In addition, two local search algorithms a 2-Opt Hill Climb and Simulated Annealing Algorithm are explored. These local search algorithms are ideal for finding low error solutions in limited time, although no performance bounds are guaranteed beyond what is expected from the Greedy Algorithms that initially seed them.

PROBLEM DEFINITION

The symmetric traveling salesman optimization problem is formalized as follows:
Given a connected undirected graph G consisting of n nodes,$\{v_0, \ldots, v_{n-1}\}$, with edge weights d_{ij} between nodes i and j. Find a Hamiltonian Cycle, a path P* where each node has degree 2, with minimum weight.
In this paper, nodes in a 2 dimensional space were considered: $v_i = \begin{bmatrix} v_{ix} \\ v_{iy} \end{bmatrix} \in \mathbb{R}^2 \quad \forall i \in [0, n-1]$
With edge weights calculated using the Euclidean distance: $e_{ij} = ||v_i, v_j||_2 = \sqrt{(v_{ix} - v_{jx})^2 + ((v_{iy} - v_{jy}^2)} \quad \forall i \neq j, i \in [0, n-1], j \in [0, n-1]$
The solution is an element in the set of vertex sequences which are Hamiltonian Cycles given by:
$\mathcal{H} = \{(v_i)_{i=0}^{n-1} : v_0 = v_{n-1}, v_i \neq v_j \quad \forall i, j \in [0, n-1]\}$
The symmetric TSP is given by following optimization problem:
$P^* = (v_i^*)_{i=0}^{n-1} = \underset{(v_i)_{i=0}^{n-1} \in \mathcal{H}}{argmin} \quad \sum_{i=0}^{n-2} ||v_i, v_{i+1}|| + ||v_0, v_{n-1}||$

RELATED WORK

ALGORITHMS

Algorithm 1 BnB($\{v_0, \ldots, v_{n-1}\}$): Find minimum cost Hamiltonian Cycle for euclidean distances

Data: $\{v_0, \ldots, v_{n-1}\}$ set of 2-D points
for all Unordered Pairs $\{i, j\}$ **do**
  Construct edge $e = (v_i, v_j, d_{ij})$
  Add e to list E of edges in increasing weight order: $E = E \cup \{e\}$
end for

Algorithm 2 2-Opt_HC($\{v_0, \ldots, v_{n-1}\}$): Approximate the minimum cost Hamiltonian Cycle for euclidean distances using a Hill Climbing local search algorithm with 2-Opt exchange Neighborhood Creation

Data: $\{v_0, \ldots, v_{n-1}\}$ set of 2-D points
for all Unordered Pairs $\{i, j\}$ **do**
  Construct edge $e = (v_i, v_j, d_{ij})$
  Add e to list E of edges in increasing weight order: $E = E \cup \{e\}$
end for
while Unassigned nodes in v **do**
  Assign Nodes to Route based on Greedy Nearest Neighbor implementation
end while
for i = 1 to length(Route Matrix) **do**
  for j = i+1 to length(Route Matrix) **do**
    reverse route[i] to route[j] and add it to newroute[i] to newroute[j]
    if cost(newroute) < cost(route) **then**
      route ⟵ newroute
    end if
  end for
end for

Algorithm 3 Sim_Anneal($\{v_0, \ldots, v_{n-1}\}$): Approximate the minimum cost Hamiltonian Cycle for euclidean distances using a Hill Climbing local search algorithm with 2-Opt exchange Neighborhood Creation

Data: $\{v_0, \ldots, v_{n-1}\}$ set of 2-D points
Current_Route: $\{c_0, \ldots, c_{n-1}\}$ Set of Location Nodes denoting the current route for annealing
Best_Route: $\{b_0, \ldots, b_{n-1}\}$ Set of Location Nodes denoting the best route calculated so far for annealing
Temperature: T Current Annealing Temperature used
Cooling Ratio: α Ratio used to cool the temperature as Simulated Annealing is run
for all Unordered Pairs $\{i, j\}$ **do**
  Construct edge $e = (v_i, v_j, d_{ij})$
  Add e to list E of edges in increasing weight order: $E = E \cup \{e\}$
end for
while Unassigned nodes in v **do**
  Assign Nodes to Route based on Greedy Nearest Neighbor implementation
end while
while Temperature ≥ Ending Temperature **do**
  Generate Random 2 Exchange Permutation
  if Current Solution Cost > Neighbor Route Cost **then**
    Update Current Route
  else
    Calculate Probability Using Current Temperature
    if Probability > Randomly Generated Probability **then**
      Update Current Route
    end if
  end if
  if Current Cost *leq* Best Cost **then**
    Update Best route
  end if
end while

EMPIRICAL EVALUATION

DISCUSSION

CONCLUSION

REFERENCES

[1] Gilbert Laporte. 1992. The traveling salesman problem: An overview of exact and approximate algorithms. *European Journal of Operational Research* 59, 2 (1992), 231–247. https://doi.org/10.1016/0377-2217(92)90138-y