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. The results derived showed that the local search algorithms were extremely adept at providing relatively good solutions even in the presence of large data sets, however they struggled to consistently return optimal solutions even for the smaller data-sets. Of the two local search algorithms explored, the simulated annealing algorithm provided clear benefits especially for larger data sets which contain a higher number of local optima which can frequently trap other algorithms in a non optimal neighborhood.

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

EMPIRICAL EVALUATION

DISCUSSION

Branch-and-Bound

Approximation Algorithm

Local Search The 2-Opt Exchange and Simulated Annealing algorithms each performed relatively well for the majority of data-sets based on algorithm run-time versus performance. For each algorithm instance, a short algorithm runtime was able to produce results which were in most cases within 5% of the optimal solution. Each algorithm setup used a simple Greedy approximation algorithm as the initial path for local search. When testing different initial paths, the Greedy path proved to provide a good mix between accuracy and execution time as a strong initial solution was presented quickly by the algorithm, allowing more local search iterations per time-frame. The Simulated Annealing provided a clear benefit over the 2-opt exchange hill climbing setup however, which was due mainly to the ability of the algorithm to consider a broader neighborhood than the strict neighborhood that the 2-Opt exchange argument considered. To more clearly identify this, the α value was varied with a clear decrease in performance observed when the α value was deviated far from the eventual selection of .98 in either direction. This α value proved to have a reasonable affect on the annealing time-frame which allowed the algorithm to consider the entire search space thoroughly enough while still reaching a reasonable solution in a relatively short period of time. The 2-Opt algorithm struggled with a smaller neighborhood as only 2-Opt exchanges starting from all Greedy solutions were considered, and only those neighborhoods who provided a clear one to one improvement over the current best were used. The power of the Simulated Annealing

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, ..., 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 ∪ {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 array (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, ..., v_{n-1}} set of 2-D points
Current_Route: {c_0, ..., c_{n-1}} Set of Location Nodes denoting the current route for annealing
Best_Route: {b_0, ..., 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 ∪ {e}
end for
while Unassigned nodes in v do
    Assign next node in route as the remaining node with the shortest distance between itself and the current node
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 ≤ Best Cost then
        Update Best Route
    end if
end while
```

setup becomes clear when looking at both of these local search algorithms as a similar 2-Opt Exchange neighborhood is considered in both, however the ability of Simulated Annealing to allow worse routes temporarily with the hope that they eventually lead to a more desireable solution proved to be key.

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