1 INTRODUCTION

The Traveling Salesman Problem (TSP) is an NP-Complete computational problem which has been studied extensively for several decades [**?**]. 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 subproblems 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.

PROBLEM DEFINITION

The symmetric traveling salesman optimization problem is formalized as follows:
Given a connected undirected graph G consisting of n nodes,$\{v_0, \dots, v_{n-1}\}$, 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($\{v_0, \dots, v_{n-1}\}$): Find minimum cost Hamiltonian Cycle for euclidean distances

```
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
```

Algorithm 2 2-Opt Local Search($\{v_0, \dots, v_{n-1}\}$): Approximate the minimum cost Hamiltonian Cycle for euclidean distances

```
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 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
```

EMPIRICAL EVALUATION

DISCUSSION

CONCLUSION