# 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 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,$\{v_0, \ldots, v_{n-1}\}$ , 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:
$P^* = (v_i^*)_{i=0}^{n-1} = \quad \underset{(v_i)_{i=0}^{n-1} \in \mathcal{H}}{argmin} \quad \sum_{i=0}^{n-2} ||v_i, v_{i+1}|| + ||v_0, v_{n-1}||$

## RELATED WORK

## ALGORITHMS

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

```
Data: {v_0, ..., 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( $\{v_0, \ldots, v_{n-1}\}$ ): 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
```

**Table 1: Approx Algorithm Performance Table**

| Instance | Time[s] | Solution Quality | Relative Error |
|---|---|---|---|
| Atlanta | 0.00 | 2488307 | 0.2418 |
| Berlin | 0.02 | 10114 | 0.3410 |
| Boston | 0.00 | 1107063 | 0.2390 |
| Champaign | 0.02 | 64760 | 0.2302 |
| Cincinnati | 0.00 | 318227 | 0.1449 |
| Denver | 0.03 | 129206 | 0.2865 |
| NYC | 0.02 | 1927253 | 0.2393 |
| Philadelphia | 0.00 | 1697409 | 0.2159 |
| Roanoke | 0.27 | 796030 | 0.2145 |
| SanFrancisco | 0.03 | 1085013 | 0.3392 |
| Toronto | 0.06 | 1652074 | 0.4046 |
| UKansasState | 0.00 | 70318 | 0.1168 |
| UMissouri | 0.05 | 170427 | 0.2842 |

**Table 2: BnB Algorithm Performance Table**

| Instance | Time[s] | Solution Quality | Relative Error |
|---|---|---|---|
| Atlanta | 0.02 | 2003763 | 0.0000 |
| Berlin | 0.55 | 8087 | 0.0723 |
| Boston | 402.47 | 958728 | 0.0730 |
| Champaign | 42.53 | 60181 | 0.1432 |
| Cincinnati | 52.53 | 277952 | 0.0000 |
| Denver | 570.83 | 112944 | 0.1246 |
| NYC | 1.55 | 1783236 | 0.1467 |
| Philadelphia | 177.14 | 1482811 | 0.0622 |
| Roanoke | 541.27 | 757720 | 0.1560 |
| SanFrancisco | 6.59 | 857727 | 0.0587 |

**Table 3: LS1 Algorithm Performance Table**

| Instance | Time[s] | Solution Quality | Relative Error |
|---|---|---|---|
| Atlanta | 0.01 | 2003763 | 0.0000 |
| Berlin | 0.41 | 8087 | 0.0723 |
| Boston | 0.04 | 999953 | 0.1191 |
| Champaign | 0.54 | 61010 | 0.1589 |
| Cincinnati | 5.34 | 214510 | 0.2282 |
| Denver | 2.84 | 116743 | 0.1624 |
| NYC | 0.80 | 1783236 | 0.1467 |
| Philadelphia | 0.05 | 1498008 | 0.0731 |
| Roanoke | 19.65 | 782292 | 0.1935 |
| SanFrancisco | 2.54 | 857727 | 0.0587 |
| Toronto | 6.34 | 1218693 | 0.0362 |
| UKansasState | 2.09 | 42791 | 0.3204 |
| UMissouri | 1.84 | 153554 | 0.1571 |

**Table 4: LS2 Algorithm Performance Table**

| Instance | Time[s] | Solution Quality | Relative Error |
|---|---|---|---|
| Atlanta | 0.03 | 2003763 | 0.0000 |
| Berlin | 9.51 | 7712 | 0.0225 |
| Boston | 10.35 | 908113 | 0.0163 |
| Champaign | 11.01 | 53896 | 0.0238 |
| Cincinnati | 0.01 | 280282 | 0.0084 |
| Denver | 7.67 | 109965 | 0.0949 |
| NYC | 9.36 | 1666127 | 0.0714 |
| Philadelphia | 9.50 | 1396495 | 0.0004 |
| Roanoke | 7.14 | 757971 | 0.1564 |
| SanFrancisco | 9.31 | 845412 | 0.0435 |
| Toronto | 8.63 | 1212031 | 0.0305 |
| UKansasState | 0.00 | 63664 | 0.0111 |
| UMissouri | 13.48 | 147505 | 0.1115 |

## 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), 231âĂŞ247. https://doi.org/10.1016/0377-2217(92)90138-y