# A Multi-Start Iterated Local Search Algorithm for the Bottleneck Travelling Salesman Problem

*A Project Report submitted in partial fulfilment of the requirements*

*for the degree of B.Tech*

*by*


Viknesh Rajaramon
(Roll No: COE18B060)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
INDIAN INSTITUTE OF INFORMATION TECHNOLOGY,
DESIGN AND MANUFACTURING, KANCHEEPURAM


May 2022

# Certificate

I, **Viknesh Rajaramon**, with Roll No: **COE18B060** hereby declare that the material presented in the Project Report titled **A Multi-Start Iterated Local Search Algorithm for the Bottleneck Travelling Salesman Problem** represents original work carried out by me in the **Department of Computer Science and Engineering** at the **Indian Institute of Information Technology, Design and Manufacturing, Kancheepuram** during the year **2022**. With my signature, I certify that:

- I have not manipulated any of the data or results.
- I have not committed any plagiarism of intellectual property. I have clearly indicated and referenced the contributions of others.
- I have explicitly acknowledged all collaborative research and discussions.
- I have understood that any false claim will result in severe disciplinary action.
- I have understood that the work may be screened for any form of academic misconduct.

Date: **10 May 2022** Student's Signature

In my capacity as supervisor of the above-mentioned work, I certify that the work presented in this Report is carried out under my supervision, and is worthy of consideration for the requirements of project work during the period January 2022 to May 2022.

Advisor's Name: **Dr. Venkatesh Pandiri** Advisor's Signature

# *Abstract*

The bottleneck travelling salesman problem (BTSP) is a variation of the well-known travelling salesman problem (TSP) in which the goal is to identify a Hamiltonian circuit on a graph with lowest maximum edge cost among its constituent edges. The BTSP finds application in the area of workforce planning and in minimizing make-span in a two-machine flow shop with no-wait-in-process. A multi-start iterated local search method for the BTSP is proposed in this paper. As part of this approach, two local search algorithms have been developed - one based on insertion and the other based on modified 2-opt moves. Performance of the suggested approach is investigated by using the standard TSPLIB library's benchmark instances. The suggested approach's effectiveness is demonstrated through computational results and their interpretation.

# *Acknowledgements*

I express my sincere gratitude to **Dr. Pandiri Venkatesh**, IIITDM Kancheepuram for being my mentor and guiding me in this project. His suggestions and advice have never failed to leave me with ideas to explore. Every interaction of ours has made this project better.

I would like to offer my thanks to my professors at IIITDM Kancheepuram for piquing my fascination in their courses. I owe special thanks to **Dr. Sadagopan N** for helping nurture my inchoate fascination for theoretical computer science to perennial passion today.

I am deeply grateful to my friends for helping me and for offering me with the best social setting I could wish for. I would like to thank my family for providing me with a conductive environment to pursue and nurture my interests which are reflected in but not limited in this project.

# Contents

# List of Figures

# List of Tables

# Abbreviations

| | |
|---|---|
| **TSP** | **T**ravelling **S**alesman **P**roblem |
| **TSPLIB** | **T**ravelling **S**alesman **P**roblem **LIB**rary |
| **BTSP** | **B**ottleneck **T**ravelling **S**alesman **P**roblem |
| **MTSP** | **M**aximum **T**ravelling **S**alesman **P**roblem |
| **MSTSP** | **M**aximum **S**catter **T**ravelling **S**alesman **P**roblem |
| **VRP** | **V**ehicle **R**outing **P**roblem |
| **NP** | **N**ondeterministic **P**olynomial |
| **BST** | **B**inary **S**earch (based) **T**hreshold |
| **HSCS** | **H**ybrid **S**equential **C**onstructive **S**ampling |
| **GA** | **G**enetic **A**lgorithm |
| **HGA** | **H**ybrid **G**enetic **A**lgorithm |
| **ILS** | **I**terated **L**ocal **S**earch |
| **MS-ILS** | **Multi-S**tart **I**terated **L**ocal **S**earch |

# Chapter 1

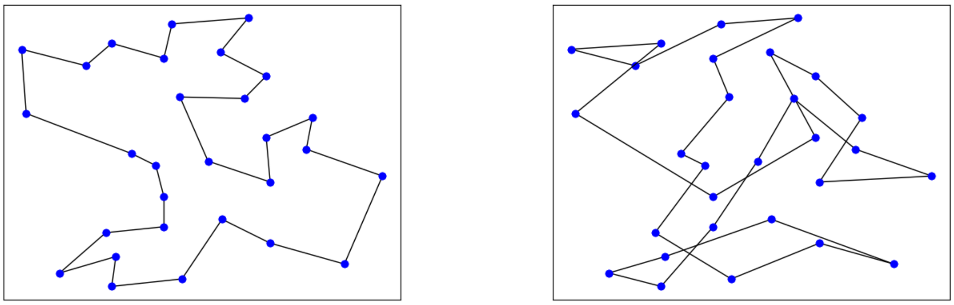
# Introduction

## 1.1 Introduction

The Bottleneck Travelling Salesman Problem (BTSP) is a version of the Travelling Salesman Problem (TSP). The TSP searches for a Hamiltonian cycle of lowest length across a set of nodes (also called cities). Reducing the overall distance travelled by the salesman is the goal of the TSP. BTSP, like TSP, is looking for a Hamiltonian circuit. Of contrast to the TSP, the aim in BTSP is to decrease the edge length that has the longest length of all the edges in the Hamiltonian circuit. The difference between the TSP and the BTSP on a TSPLIB instance **bays29** having 29 nodes, is shown in Figure 1.1. We can clearly see that the BTSP tries to minimize the maximum distance by making any two consecutive nodes as close as possible whereas the TSP tries to minimize the overall distance travelled by the salesman. The Maximum Scatter Travelling Salesman Problem (MSTSP) is a problem similar to the BTSP where the aim is to maximise the shortest edge length in a Hamiltonian cycle.

## 1.2 Problem Statement

Given an undirected edge-weighted complete graph $G = (V(G), E(G))$, where $V(G) = \{1, 2, 3, \ldots, n\}$ is the set of nodes and $E(G) = \{(u, v) | u, v \in V(G)\}$. Define a binary



(A) Solution of TSP (B) Solution of BTSP

FIGURE 1.1: Difference between TSP and BTSP using instance **bays29**

variable $x_{uv}$ as follows:

$$x_{uv} = \begin{cases} 1, & \text{path exists between nodes } u \text{ and } v \\ 0, & \text{otherwise} \end{cases} \tag{1.1}$$

Take $d_{uv} > 0$ to be the distance between node $u$ and node $v$. Then the BTSP can be represented as the following mathematical model:

$$\text{Minimize} \max_{(u,v)\in E(G)} d_{uv}x_{uv} \tag{1.2}$$

subject to:

$$\sum_{(u,v)\in E(G)} x_{uv} = \sum_{(v,w)\in E(G)} x_{vw} = 1, \qquad \forall v \in V(G), \tag{1.3}$$

$$\sum_{u\in S}\sum_{v\in S} x_{uv} \leq |S| - 1, \qquad \forall S \subset V(G) \tag{1.4}$$

- The objective function for the BTSP is given by Equation (1.2) and it minimizes the maximum edge cost.
- Equation (1.3) denotes the in-degree and out-degree constraints as every node must have exactly one incoming edge and one outgoing edge.
- Equation (1.4) ensures that no proper subset S can form a sub-tour, so that the solution returned is a single tour and not a union of smaller tours.

# Chapter 2

# Literature Survey

## 2.1 Studies on Bottleneck TSP

This section consists of the details of various research by various authors relating to our study of interest and discuss them.

In 1964, Gilmore and Gormory [1–7] presented a special instance of the BTSP. In 1971, Gabovic et al. [8] introduced the general BTSP. The BTSP finds application in the area of workforce planning [6] and in minimizing make-span in a two-machine flow shop with no-wait-in-process [9]. The Maximum Scatter Travelling Salesman Problem (MSTSP) is a problem that is similar to the BTSP in that it aims to maximise the least edge cost in a Hamiltonian circuit.

The BTSP is an NP-hard problem, and is hence very difficult to solve by conventional methods. All the known exact algorithms for solving the BTSP are enumerative in nature. Many studies proposed approximate algorithms to solve the BTSP. there doesn't exist a polynomial time $\epsilon$-approximation algorithm for any $\epsilon > 0$ [10–12]. However, polynomial time algorithms for various special cases exist with guaranteed performance ratios. For example, when the $\tau$-triangle inequality (i.e., $c_{ij} \leq \tau(c_{ik} + c_{jk})\ \forall i, j, k \in V$ and $\tau \geq \frac{1}{2}$) is satisfied by the edge cost, a $2\tau$-approximate solution may be achieved in polynomial time [3], and this appears to be the best performance bound for this problem. Even if the edge costs are satisfied by the $\tau$-triangle inequality for $\tau > \frac{1}{2}$ [3], there is no $2\tau - \epsilon$ approximation procedure for the BTSP unless P = NP. However, no meta-heuristic approach has been developed for solving this problem so far.

A branch and bound method was presented by Garfinkel and Gilbert [13]. They presented computational findings on random problems with nodes ranging from 10 to 100 using a constructive heuristic. In many situations, the algorithm yielded optimal results. The solution quality looked to be deteriorating as the instance size increased from 10 to 100 in one set of problems.

A basic Binary Search based Threshold (BST) method was proposed by Ramakrishna et al. [14] using the 2-max bound [5] as a beginning lower bound and the objective function value of the closest neighbour heuristic as an upper bound. From this, we can obtain the 2-max bound. The optimality of some instances was not proved due to the poor 2-max lower bound.

A hybrid sequential constructive sampling (HSCS) algorithm was implemented by Ahmed [15] which incorporates a combined mutation operator to the sequential constructive sampling algorithm. To diversify the population of chromosomes in the Genetic Algorithm (GA), mutation operation [16] was used. The insertion operator (pick a node and insert it in a random place), inversion operator (select two places along the length of chromosome and reverse the sub-tour between those places), and reciprocal exchange operator (select two nodes randomly and swap them) are the most often used mutation operators for TSP.

Ahmed [17] employs a basic genetic Algorithm (GA) with sequential constructive crossover [15] to get a heuristic solution. 2-opt search and an additional local search method was incorporated into the standard GA to enhance the results. The GA starts with a set of chromosomes termed the starting population, and then applies three operations to get a heuristically optimum solution: reproduction/selection, crossover, and mutation. To prevent the solution from being stuck at local minimum, a certain percentage of the population was replaced at random with a set of new chromosomes.

# Chapter 3

# Methodology

## 3.1 Multi-Start Iterated Local Search Based Algorithm

This section begins with a quick overview of the iterated local search algorithm before diving into the specifics of the proposed algorithm for solving the BTSP.

Iterated Local Search (ILS) is a meta-heuristic that improves the quality of a single solution iteratively. ILS, according to [18], [19], offers a number of desirable characteristics, including being simple to apply, resilient, and extremely effective. Initial solution generation, local search (exploit the solution space), acceptance criteria and perturbation technique (explore the solution space) are the four primary components of the ILS. An iterative procedure follows, starting with an initial solution. During each iteration, the current solution is first subjected to the local search algorithm in order to find the local optimum solution. The newly acquired locally optimum solution may then replace the existing solution depending on the acceptance criteria. To escape that locally optimum solution, a perturbation technique is used on the current solution, resulting in a perturbed solution. The perturbed solution is used at the current solution for the following iteration.

Two commonly used acceptance criterion's are:

- Always replace the existing solution with newly obtained solution. This results in a random-walk method.
- Replace the existing solution with the new solution if and only if it is better than the current solution. This results in a improvement type of of method.

ILS has been used to solve a variety of optimization problems and has proven to be effective to other methods, e.g., [20], [21], [22], [23], [24]. Algorithm 1 contains the pseudo-code for the basic ILS.

**Algorithm 1:** Pseudo-code for basic ILS

**Input:** ILS parameters
**Output:** Best solution found

$T \leftarrow Initial\_Solution();$

**while** *Termination condition not satisfied* **do**
  $T_1 \leftarrow Local\_Search(T);$
  $T \leftarrow Acceptance\_Criteria(T, T_1, history);$
  $T \leftarrow Perturbation\_Procedure(T);$

**return** *best*;

## 3.2 Proposed theory

The proposed multi-start iterated local search (MS-ILS) for the BTSP is an extension of ILS that restarts the ILS numerous times, each time starting with a new solution provided by the initial solution generation technique. The multi-start approach was chosen to avoid unfruitful iterations from consuming time. The search gave better results when it was restarted with a newly created initial solution. In the following subsections, the components of the proposed approach are addressed.

### 3.2.1 Solution Encoding and Fitness

A solution is encoded in the MS-ILS by a linear permutation of nodes, with the first node always occupying the first place. The redundancy in representation is reduced by limiting the first node to the first place. Please keep in mind that none of the MS-ILS components may change the first node's location.

Since BTSP is a minimization problem, the fitness function is defined as the reciprocal of the objective function described in the equation (1.2). Solution with a higher fitness function value (lower maximum edge cost) is preferred over a solution with a lower fitness value (higher maximum edge cost).

### 3.2.2 Initial Solution Generation

The initial solution generation begins with a random selection of two nodes, followed by an iterative process. Every iteration involves randomly selecting a node and inserting it between the nodes with the highest cost edge. All the nodes are added into the tour by continuing this process.

### 3.2.3 Local Search Procedure

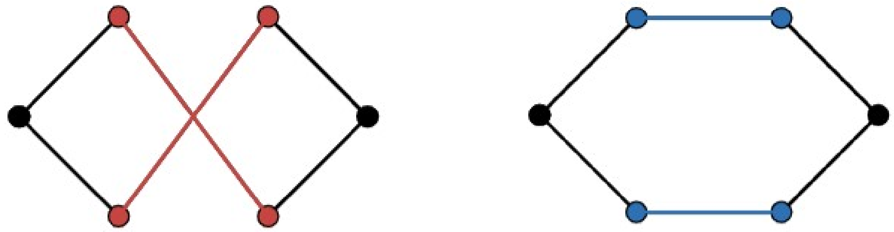
Local search and perturbation processes are critical in the ILS because they govern how the search behaves. Local search tries to exploit the neighbourhood of the current solution in order to find a better solution. The suggested MS-ILS local search technique consists of two heuristics, $h_1$ and $h_2$, each handling two situations. In the first scenario (hence referred to as case 1), there is only one maximum edge cost, however in the second scenario (hence referred to as case 2), there might be many edges with maximum edge cost. For case 1, the two heuristics attempt to minimize the maximum edge cost. For case 2, the two heuristics attempt to minimize the number of edges with maximum edge cost. As a result, the best solutions for these two scenarios are determined. Note that the maximum edge cost cannot be reduced until and unless all the edges with maximum edge cost are replaced, which is why case 2 is handled differently from case 1.

#### 3.2.3.1 $\mathbf{h_1}$: Insertion between the nodes of maximum edge cost

To reduce the maximum edge cost in case 1, this heuristic inserts a node between the nodes of maximum edge cost. As part of this heuristic, each node must be tested for insertion between the nodes with maximum edge cost, with the best of all the results being accepted. In case 2, the heuristic evaluates each edge with maximum edge cost in the solution one by one, in the sequence in which they appear. To limit the number edges with maximum edge cost, each node is put one by one between the nodes of edge under examination, and the best of all the resultant solutions is accepted. If the number of edges with maximum edge cost are reduced, this heuristic ends and $h_2$ begins. Otherwise, next edge with maximum edge cost is considered

#### 3.2.3.2 $\mathbf{h_2}$: Maximum cost edge centric 2-opt move

Two edges are deleted from the tour in a 2-opt move and the resultant two paths are linked through two other edges. After attempting every pair of edges in the tour, the best tour is obtained. Figure 3.1 shows an example of a 2-opt move. The two red coloured edges are deleted from the path in this illustration, and the two blue coloured edges are utilised to reconstruct it. The suggested heuristic is a modified version of the 2-opt move, with the maximum cost edge always being one of the edges to be eliminated. To reduce the maximum edge cost in case 1, every other edge must be attempted with the maximum edge for removal, and two new edges must be placed in their place, according to our heuristic. Accept the move that results in the greatest reduction in the maximum edge cost. In case 2, the heuristic operates in the same way as heuristic $h_1$, with the control being handed to $h_1$ after the number of edges with maximum edge cost is reduced. Switching control between one another as soon as the solution improves, rather than employing $h_1$ and $h_2$ sequentially until no improvement is achieved, produces a better final solution. As a result, $h_1$ and $h_2$ are used interchangeably.



(A) Before 2-opt move (B) After 2-opt move

FIGURE 3.1: Example of 2-opt move

### 3.2.4 Acceptance Criteria

The chosen acceptance criteria involves comparison of the quality (fitness) of the solution provided by the local search with the solution generated before this procedure was used. At all occasions, a solution with better fitness value is accepted. The equal fitness solution is accepted when the maximum edge cost is valued at multiple edges and the number of edges with that cost decreases. When all of the previous scenarios fail, the perturbation technique is used, which may result in a better or poorer fitness solution. The newly returned solution is used for the search procedure in the next iteration.

### 3.2.5 Perturbation Procedure

The purpose of the perturbation technique is to transfer the search to undiscovered parts of the search space by perturbing the current locally optimal solution which would help provide a changed beginning solution to the local search. In essence, local search strategies are used for exploitation, whereas perturbation approaches are used for exploration. This perturbation approach is used if none of the heuristics $h_1$ and $h_2$ are able to enhance the solution in terms of maximum edge cost and the number of edges with maximum edge cost. A destroy and repair mechanism is employed as part of this approach. With a probability $P_{pert}$, each node on the tour is removed. All of the removed nodes are iteratively inserted back into the tour in the same that the initial solution generating technique outlined in Section 3.2.2. The pseudo-code for the technique to perturb a solution is provided in Algorithm 2.

Algorithm 3 provides the pseudo-code for the proposed MS-ILS approach for the BTSP.

**Algorithm 2:** Pseudo-code for perturbing a solution

**Input:** A solution $T$

**Output:** A perturbed solution $T_1$

```
Function Perturbation_Procedure(T):
    foreach node c in tour of T do
        Generate a random number 0 ≤ p ≤ 1
        if p < P_pert then
            Add c to a set of unassigned nodes
        else
            Copy c to tour of T_1
    foreach node c in the set of unassigned nodes in some random order do
        Follow the procedure described in Section 3.2.2 to insert c into tour of T_1
return T_1;
```

**Algorithm 3:** Pseudo-code for the proposed MS-ILS approach to solve the BTSP

**Input:** Set of parameters for the MS-ILS and a BTSP instance

**Output:** Best solution found

```
F(best) = −∞
for st = 1 to N_rst do
    T ← Initial_Solution();
    while Termination condition not satisfied do
        /* Apply heuristic h_1 */
        T_1 ← Apply_heuristic_h_1(T);
        if F(T_1) > F(T) then
            T ← T_1;
        else if F(T_1) > F(T) then
            if no. of edges with F(T) decreased then
                T ← T_1;
        /* Apply heuristic h_2 */
        T_1 ← Apply_heuristic_h_2(T);
        if F(T_1) > F(T) then
            T ← T_1;
        else if F(T_1) > F(T) then
            if no. of edges with F(T) decreased then
                T ← T_1;
        /* Dealing with the best solution and local optimum solution */
        if F(T) > F(best) then
            best ← T;
        else if F(T_1) < F(T) then
            T_1 ← Perturbation_Procedure();
return best;
```

# Chapter 4

# Work Done

## 4.1 Dataset Acquisition

TSPLIB instances each with different number of nodes and coordinates (latitude and longitude) are downloaded from the standard TSPLIB library[1]. Each instance consists of data with different edge types (GEO, EUC_2D, etc). Every one of these instances is converted to a $n \times n$ distance matrix where the distance between any two nodes is an integer. The diagonal elements in the $n \times n$ distance matrix is represented as $\infty$.

[1] http://elib.zib.de/pub/mp-testdata/tsp/tsplib/tsplib.html

```
NAME: burma14
TYPE: TSP
COMMENT: 14-Staedte in Burma (Zaw Win)
DIMENSION: 14
EDGE_WEIGHT_TYPE: GEO
EDGE_WEIGHT_FORMAT: FUNCTION
DISPLAY_DATA_TYPE: COORD_DISPLAY
NODE_COORD_SECTION
   1  16.47       96.10
   2  16.47       94.44
   3  20.09       92.54
   4  22.39       93.37
   5  25.23       97.24
   6  22.00       96.05
   7  20.47       97.02
   8  17.20       96.29
   9  16.30       97.38
  10  14.05       98.12
  11  16.53       97.38
  12  21.52       95.59
  13  19.41       97.13
  14  20.09       94.55
EOF
```

FIGURE 4.1: **burma14** instance from the standard TSPLIB library

| Node | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | ∞ | 153 | 510 | 706 | 966 | 581 | 455 | 70 | 160 | 372 | 157 | 567 | 342 | 398 |
| 2 | 153 | ∞ | 422 | 664 | 997 | 598 | 507 | 197 | 311 | 479 | 310 | 581 | 417 | 376 |
| 3 | 510 | 422 | ∞ | 289 | 744 | 390 | 437 | 491 | 645 | 880 | 618 | 374 | 455 | 211 |
| 4 | 706 | 664 | 289 | ∞ | 491 | 265 | 410 | 664 | 804 | 1070 | 768 | 259 | 455 | 310 |
| 5 | 966 | 997 | 744 | 491 | ∞ | 400 | 514 | 902 | 990 | 1261 | 947 | 418 | 499 | 636 |
| 6 | 581 | 598 | 390 | 265 | 400 | ∞ | 168 | 522 | 634 | 910 | 593 | 19 | 635 | 239 |
| 7 | 455 | 507 | 437 | 410 | 514 | 168 | ∞ | 389 | 482 | 757 | 439 | 163 | 284 | 232 |
| 8 | 70 | 197 | 491 | 664 | 902 | 522 | 389 | ∞ | 154 | 406 | 133 | 508 | 124 | 355 |
| 9 | 160 | 311 | 645 | 804 | 990 | 634 | 482 | 154 | ∞ | 276 | 43 | 623 | 273 | 498 |
| 10 | 372 | 479 | 880 | 1070 | 1261 | 910 | 757 | 406 | 276 | ∞ | 318 | 898 | 358 | 761 |
| 11 | 157 | 310 | 618 | 768 | 947 | 593 | 439 | 133 | 43 | 318 | ∞ | 582 | 633 | 464 |
| 12 | 567 | 581 | 374 | 259 | 418 | 19 | 163 | 508 | 623 | 898 | 582 | ∞ | 315 | 221 |
| 13 | 342 | 417 | 455 | 499 | 635 | 284 | 124 | 273 | 358 | 633 | 315 | 275 | ∞ | 247 |
| 14 | 398 | 376 | 211 | 310 | 636 | 239 | 232 | 355 | 498 | 761 | 464 | 221 | 247 | ∞ |

FIGURE 4.2: Distance matrix for **burma14** instance

## 4.2 Choosing the Correct Parameters

The proposed algorithm comprises of three parameters namely, $P_{pert}$, $N_{rst}$ and $RUNS$ which need to be optimized in order to derive better results. $P_{pert}$ denotes the probability with which a node is removed during the perturbation procedure 3.2.5. $N_{rst}$ represents the number of times the algorithm is restarted after being stuck in local optimal solution. $RUNS$ denotes the number of times the algorithm is run on the same instance, each time with a random seed value.

### 4.2.1 Choosing $\mathbf{P_{pert}}$

From Figure 4.3, we can clearly see that both of the TSPLIB instances **att48** and **pr76** achieve a minimum value of 544.40 and 4543.60 on average when $P_{pert} = 0.2$.

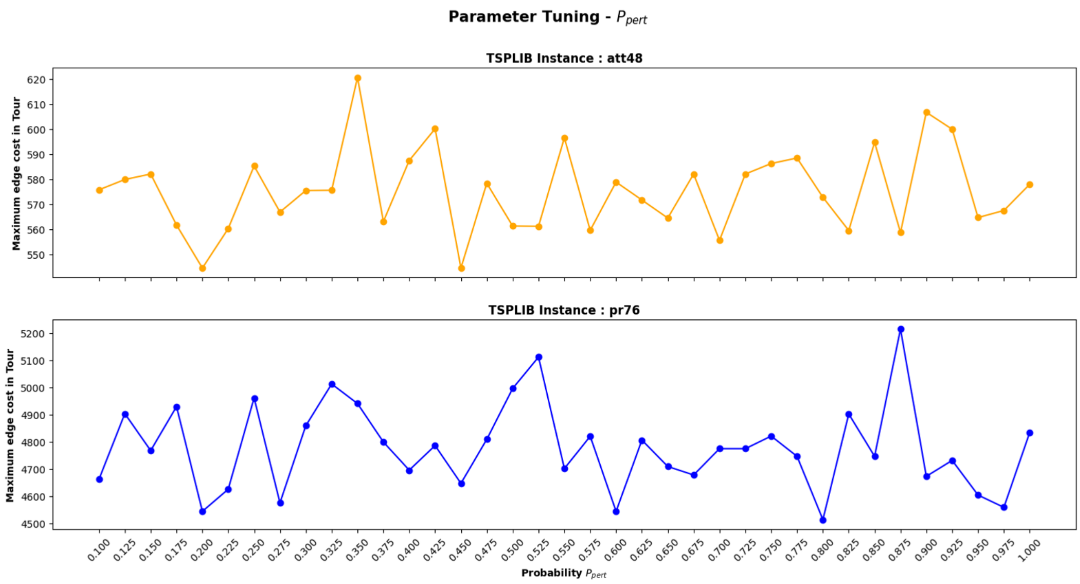



FIGURE 4.3: Parameter Tuning - Perturbation Probability ($P_{pert}$)

### 4.2.2 Choosing $\mathbf{N_{rst}}$

From Figure 4.4, we can clearly see that both of the TSPLIB instances **gr24** and **eil76** achieve a minimum value of 108.00 and 29.70 on average when $N_{rst} = 100$.

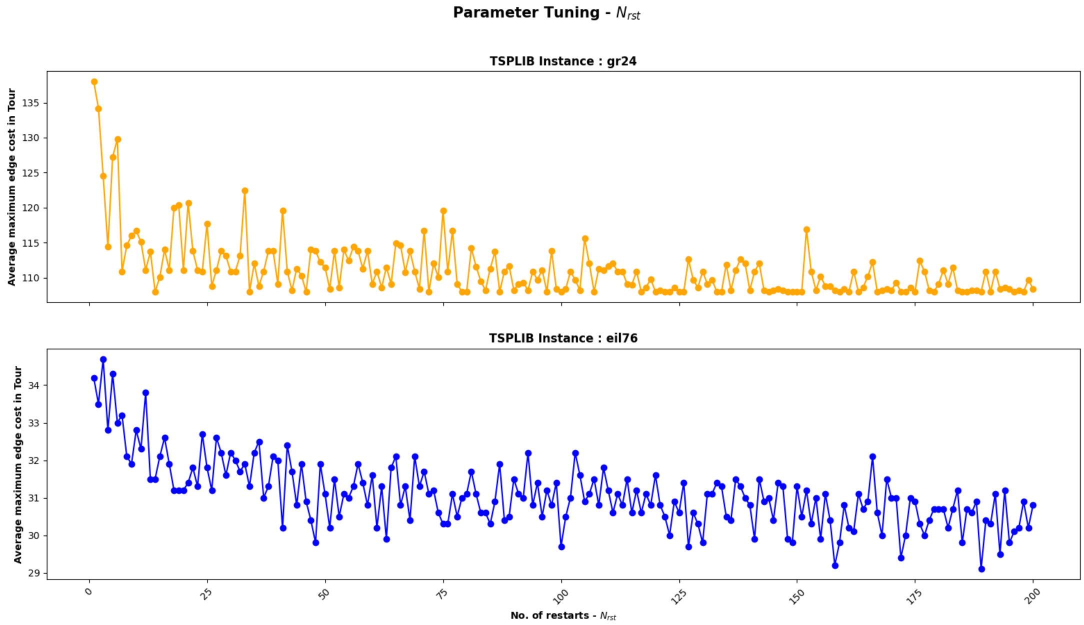



FIGURE 4.4: Parameter Tuning - No. of Restarts ($N_{rst}$)

### 4.2.3 Choosing RUNS

From Figure 4.5, we can clearly see that both of the TSPLIB instances **swiss42** and **berlin52** achieve a minimum value of 82.00 and 480.20 on average when $RUNS = 10$.

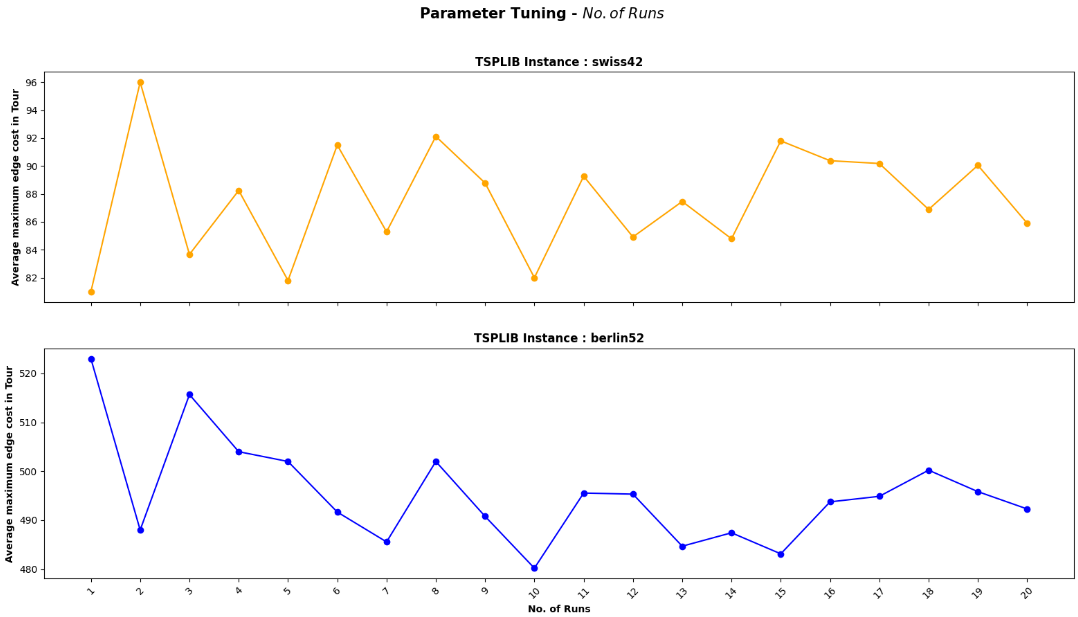



FIGURE 4.5: Parameter Tuning - No. of Runs ($RUNS$)

## 4.3 Results on few Test Instances

The proposed MS-ILS technique is run 10 times on each test instance, starting with a different seed value each time. MS-ILS is written in C++ and runs on a 2.6GHz Core-i7-10750H Linux machine with 8GB RAM. The following settings are used for all the test instances: MS-ILS is restarted 100 times, i.e., $N_{rst} = 100, P_{pert} = 0.2$. When the solution doesn't improve continuously for 100 iterations, the MS-ILS terminates.

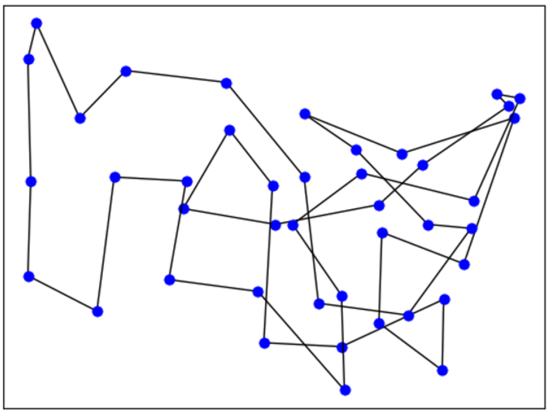
Table 4.1 shows the comparison between the proposed MS-ILS algorithm and already existing algorithms. From this table, we can clearly see that the proposed MS-ILS algorithm finds the optimal solution (in average) for all the 15 instances whereas HGA, HSCS and BST algorithms find the solution for 5, 3 and 10 instances respectively. On the basis of solution quality, the proposed MS-ILS algorithm is found to be better than BST, HSCS, and HGA algorithms. Also, on the basis of computational time, the MS-ILS algorithm is found to be the best and BST the worst.

TABLE 4.1: RESULTS OF DIFFERENT ALGORITHMS FOR SOME STANDARD TSPLIB INSTANCES

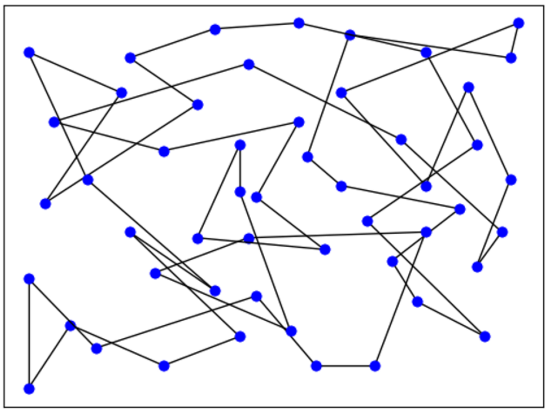
| **Instance** | **n** | **BST** | | **HSCS** | | **HGA** | | **MS-ILS($h_1 + h_2$)** | |
|---|---|---|---|---|---|---|---|---|---|
| | | **Average** | **Time** | **Average** | **Time** | **Average** | **Time** | **Average** | **Time** |
| burma14 | 14 | **418.00** | 742.38 | 422.18 | 60.57 | **418.00** | 61.80 | **418.00** | 0.02 |
| ulysses16 | 16 | **1504.00** | 834.15 | **1504.00** | 68.06 | **1504.00** | 69.44 | **1504.00** | 0.07 |
| gr17 | 17 | **282.00** | 727.00 | **282.00** | 59.31 | **282.00** | 60.52 | **282.00** | 0.03 |
| gr21 | 21 | **355.00** | 826.46 | **355.00** | 67.43 | **355.00** | 68.80 | **355.00** | 0.03 |
| ulysses22 | 22 | **1504.00** | 938.42 | 1519.04 | 76.56 | **1504.00** | 78.12 | **1504.00** | 0.13 |
| fri26 | 26 | **93.00** | 533.36 | 93.50 | 43.51 | 93.50 | 44.40 | **93.00** | 0.04 |
| brazil58 | 58 | **2149.00** | 1264.68 | 2508.54 | 103.18 | 2483.70 | 105.28 | **2149.00** | 0.13 |
| gr96 | 96 | 3491.00 | 2931.54 | 4098.90 | 239.17 | 4098.90 | 244.04 | **2807.00** | 0.36 |
| pr107 | 107 | 7053.00 | 3694.10 | 7387.40 | 301.39 | 7387.40 | 307.52 | **7050.00** | 0.40 |
| bier127 | 127 | **7486.00** | 3765.21 | 7957.80 | 307.19 | 7957.80 | 313.44 | **7486.00** | 0.43 |
| gr137 | 137 | 4282.00 | 4409.57 | 5153.63 | 359.76 | 5102.60 | 367.08 | **2132.00** | 0.86 |
| brg180 | 180 | 9000.00 | 5997.15 | 9000.00 | 489.29 | 9000.00 | 499.24 | **3500.00** | 0.96 |
| d198 | 198 | 1511.00 | 9824.34 | 1712.40 | 801.53 | 1712.40 | 817.84 | **1380.00** | 1.06 |
| gr202 | 202 | **2230.00** | 8996.92 | 2393.70 | 734.03 | 2393.70 | 748.96 | **2230.00** | 0.98 |
| d493 | 493 | **2008.00** | 81359.08 | 2045.25 | 6637.80 | 2025.00 | 6772.84 | **2008.00** | 4.82 |
| **Overall** | | 3188.00 | 8456.29 | 3423.63 | 689.92 | 3415.08 | 703.95 | **2536.62** | 0.69 |
| **NBV** | | 10 | | 3 | | 5 | | **15** | |

Table 4.2: Comparative study of MS-ILS($h_1$), MS-ILS($h_2$), MS-ILS($h_1 + h_2$)

| **Instance** | **n** | **MS-ILS($h_1$)** | | | | **MS-ILS($h_2$)** | | | | **MS-ILS($h_1 + h_2$)** | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | **Best** | **Worst** | **Average** | **Time** | **Best** | **Worst** | **Average** | **Time** | **Best** | **Worst** | **Average** | **Time** |
| burma14 | 14 | **418.00** | 422.00 | 418.40 | 0.02 | **418.00** | **418.00** | **418.00** | 0.01 | **418.00** | **418.00** | **418.00** | 0.02 |
| ulysses16 | 16 | **1504.00** | **1504.00** | **1504.00** | 0.02 | **1504.00** | **1504.00** | **1504.00** | 0.01 | **1504.00** | **1504.00** | **1504.00** | 0.07 |
| gr17 | 17 | **282.00** | **282.00** | **282.00** | 0.02 | **282.00** | **282.00** | **282.00** | 0.01 | **282.00** | **282.00** | **282.00** | 0.03 |
| gr21 | 21 | **355.00** | **355.00** | **355.00** | 0.03 | **355.00** | **355.00** | **355.00** | 0.02 | **355.00** | **355.00** | **355.00** | 0.03 |
| ulysses22 | 22 | **1504.00** | **1504.00** | **1504.00** | 0.03 | **1504.00** | **1504.00** | **1504.00** | 0.02 | **1504.00** | **1504.00** | **1504.00** | 0.13 |
| gr24 | 24 | **108.00** | **137.00** | 127.90 | 0.04 | **108.00** | 147.00 | 118.10 | 0.03 | **108.00** | **137.00** | **112.00** | 0.04 |
| fri26 | 26 | **93.00** | 98.00 | 93.90 | 0.04 | **93.00** | **93.00** | **93.00** | 0.03 | **93.00** | **93.00** | **93.00** | 0.04 |
| bayg29 | 29 | **111.00** | 150.00 | 138.70 | 0.05 | **111.00** | 148.00 | 125.80 | 0.03 | **111.00** | **127.00** | **113.20** | 0.05 |
| bays29 | 29 | **154.00** | 182.00 | 170.90 | 0.04 | **154.00** | 178.00 | **163.50** | 0.03 | **154.00** | **168.00** | 163.80 | 0.05 |
| dantzig42 | 42 | 51.00 | 69.00 | 61.10 | 0.07 | 56.00 | 64.00 | 60.50 | 0.06 | **42.00** | **60.00** | **50.70** | 0.08 |
| swiss42 | 42 | 95.00 | 108.00 | 100.10 | 0.08 | 91.00 | 106.00 | 100.60 | 0.05 | **74.00** | **100.00** | **82.00** | 0.08 |
| att48 | 48 | 636.00 | 841.00 | 716.90 | 0.06 | **519.00** | 753.00 | 608.80 | 0.06 | **519.00** | **605.00** | **552.40** | 0.10 |
| gr48 | 48 | 333.00 | 405.00 | 385.90 | 0.08 | 340.00 | 436.00 | 383.80 | 0.06 | **227.00** | **311.00** | **255.70** | 0.11 |
| hk48 | 48 | 768.00 | 959.00 | 862.40 | 0.10 | **534.00** | 801.00 | 656.00 | 0.09 | **534.00** | **768.00** | **591.20** | 0.11 |
| eil51 | 51 | 30.00 | 32.00 | 31.60 | 0.09 | 27.00 | 32.00 | 29.80 | 0.08 | **25.00** | **31.00** | **26.90** | 0.14 |
| berlin52 | 52 | 501.00 | 526.00 | 520.90 | 0.10 | **475.00** | 551.00 | 510.60 | 0.06 | **475.00** | **523.00** | **480.20** | 0.11 |
| brazil58 | 58 | **2149.00** | 2213.00 | 2155.40 | 0.11 | **2149.00** | **2149.00** | **2149.00** | 0.07 | **2149.00** | **2149.00** | **2149.00** | 0.13 |
| st70 | 70 | 48.00 | 54.00 | 51.60 | 0.19 | 47.00 | 54.00 | 51.10 | 0.12 | **36.00** | **51.00** | **46.70** | 0.25 |
| eil76 | 76 | 30.00 | 34.00 | 33.20 | 0.21 | 30.00 | 34.00 | 32.40 | 0.14 | **27.00** | **33.00** | **31.10** | 0.26 |
| pr76 | 76 | 5561.00 | 7447.00 | 6582.50 | 0.21 | **4750.00** | 6757.00 | 5744.90 | 0.15 | **4750.00** | **5224.00** | **5081.80** | 0.24 |
| gr96 | 96 | 2855.00 | 3408.00 | 3107.00 | 0.35 | **2807.00** | 3226.00 | 2986.60 | 0.22 | **2807.00** | **2807.00** | **2807.00** | 0.36 |
| rat99 | 99 | 74.00 | 80.00 | 77.00 | 0.28 | 70.00 | 78.00 | 75.50 | 0.21 | **61.00** | **75.00** | **69.00** | 0.43 |
| kroA100 | 100 | 1032.00 | 1492.00 | 1293.10 | 0.23 | 1053.00 | 1485.00 | 1321.60 | 0.19 | **634.00** | **819.00** | **727.10** | 0.53 |
| kroB100 | 100 | 1042.00 | 1475.00 | 1317.70 | 0.18 | 985.00 | 1417.00 | 1216.90 | 0.39 | **530.00** | **786.00** | **655.90** | 0.48 |
| kroC100 | 100 | 1144.00 | 1568.00 | 1350.80 | 0.21 | 1045.00 | 1481.00 | 1336.20 | 0.39 | **576.00** | **788.00** | **709.20** | 0.47 |
| kroD100 | 100 | 1235.00 | 1528.00 | 1402.20 | 0.21 | 1112.00 | 1484.00 | 1307.90 | 0.27 | **620.00** | **954.00** | **729.60** | 0.47 |
| kroE100 | 100 | 1225.00 | 1487.00 | 1401.80 | 0.27 | 1027.00 | 1588.00 | 1335.50 | 0.39 | **630.00** | **1035.00** | **775.20** | 0.45 |
| rd100 | 100 | 468.00 | 563.00 | 521.10 | 0.31 | 454.00 | 579.00 | 503.00 | 0.20 | **250.00** | **461.00** | **315.50** | 0.42 |
| eil101 | 101 | 33.00 | 36.00 | 34.20 | 0.32 | 33.00 | 36.00 | 34.50 | 0.24 | **30.00** | **35.00** | **33.20** | 0.38 |
| lin105 | 105 | 878.00 | 1068.00 | 985.80 | 0.36 | 915.00 | 1085.00 | 997.50 | 0.21 | **176.00** | **335.00** | **273.00** | 0.58 |
| pr107 | 107 | **7050.00** | **7050.00** | **7050.00** | 0.25 | **7050.00** | **7050.00** | **7050.00** | 0.19 | **7050.00** | **7050.00** | **7050.00** | 0.40 |
| gr120 | 120 | 365.00 | 403.00 | 385.30 | 0.46 | 313.00 | 417.00 | 374.90 | 0.30 | **220.00** | **370.00** | **277.20** | 0.52 |
| pr124 | 124 | 5031.00 | 6307.00 | 5591.00 | 0.43 | **3302.00** | 5841.00 | 4055.90 | 0.34 | **3302.00** | **3680.00** | **3483.20** | 0.59 |
| bier127 | 127 | **7486.00** | **7486.00** | **7486.00** | 0.42 | **7486.00** | **7486.00** | **7486.00** | 0.24 | **7486.00** | **7486.00** | **7486.00** | 0.43 |
| ch130 | 130 | 332.00 | 355.00 | 345.70 | 0.46 | 319.00 | 372.00 | 349.90 | 0.29 | **238.00** | **344.00** | **285.60** | 0.56 |
| pr136 | 136 | 5560.00 | 6298.00 | 5930.10 | 0.53 | 5469.00 | 6242.00 | 5929.30 | 0.62 | **2976.00** | **4005.00** | **3208.60** | 0.87 |
| gr137 | 137 | 3041.00 | 4161.00 | 3676.50 | 0.55 | **2132.00** | 4192.00 | 3145.10 | 0.53 | **2132.00** | **2132.00** | **2132.00** | 0.86 |
| pr144 | 144 | 4594.00 | 5390.00 | 4789.90 | 0.66 | 4350.00 | 5234.00 | 4570.60 | 0.58 | **2825.00** | **3009.00** | **2954.40** | 0.95 |
| ch150 | 150 | 339.00 | 380.00 | 363.80 | 0.57 | 326.00 | 381.00 | 354.20 | 0.47 | **240.00** | **361.00** | **303.80** | 0.68 |
| kroA150 | 150 | 1300.00 | 1583.00 | 1445.30 | 0.70 | 1235.00 | 1561.00 | 1361.30 | 0.50 | **515.00** | **1275.00** | **778.20** | 0.90 |
| kroB150 | 150 | 1242.00 | 1573.00 | 1401.10 | 0.53 | 1075.00 | 1478.00 | 1284.70 | 0.47 | **543.00** | **986.00** | **710.90** | 1.03 |
| pr152 | 152 | 6548.00 | 7152.00 | 7022.60 | 0.62 | 6374.00 | 7150.00 | 6808.10 | 0.64 | **5553.00** | **6799.00** | **5849.10** | 0.81 |
| u159 | 159 | 2608.00 | 2773.00 | 2704.10 | 0.54 | 2433.00 | 2823.00 | 2659.60 | 0.47 | **1844.00** | **2702.00** | **2221.30** | 0.87 |
| si175 | 175 | 284.00 | **289.00** | 285.80 | 0.80 | 284.00 | **289.00** | 285.80 | 0.61 | **272.00** | **289.00** | **283.50** | 1.24 |
| brg180 | 180 | **3500.00** | **3500.00** | **3500.00** | 0.54 | **3500.00** | **3500.00** | **3500.00** | 0.52 | **3500.00** | **3500.00** | **3500.00** | 0.96 |
| rat195 | 195 | 100.00 | 112.00 | 108.40 | 0.78 | 102.00 | **110.00** | 106.90 | 0.67 | **93.00** | 111.00 | **104.80** | 1.10 |
| d198 | 198 | **1380.00** | 1602.00 | 1506.10 | 0.85 | **1380.00** | **1380.00** | **1380.00** | 0.64 | **1380.00** | **1380.00** | **1380.00** | 1.06 |
| kroA200 | 200 | 1392.00 | 1566.00 | 1483.50 | 0.64 | 1353.00 | 1592.00 | 1460.50 | 0.76 | **603.00** | **1092.00** | **779.60** | 1.87 |
| kroB200 | 200 | 1404.00 | 1573.00 | 1486.10 | 0.77 | 1353.00 | 1599.00 | 1499.80 | 0.62 | **593.00** | **1199.00** | **907.40** | 1.64 |
| gr202 | 202 | **2230.00** | 2398.00 | 2251.10 | 0.72 | **2230.00** | 2250.00 | 2232.00 | 0.59 | **2230.00** | **2230.00** | **2230.00** | 0.98 |
| ts225 | 225 | 7159.00 | 7500.00 | 7265.00 | 1.23 | 7000.00 | 7500.00 | 7276.60 | 0.96 | **6708.00** | **7433.00** | **7035.20** | 1.62 |
| tsp225 | 225 | 178.00 | 193.00 | 186.40 | 1.16 | 185.00 | 196.00 | 190.00 | 0.92 | **169.00** | **192.00** | **182.00** | 1.39 |
| pr226 | 226 | 7653.00 | 8050.00 | 7842.90 | 0.95 | 7650.00 | 8150.00 | 7772.40 | 0.81 | **3250.00** | **3650.00** | **3368.30** | 2.39 |
| gr229 | 229 | 4746.00 | 6833.00 | 5641.20 | 1.22 | 4203.00 | 6284.00 | 5411.10 | 1.01 | **4027.00** | **4203.00** | **4064.40** | 1.82 |
| gil262 | 262 | 106.00 | 110.00 | 107.90 | 1.42 | 101.00 | **108.00** | 105.00 | 1.55 | **99.00** | 109.00 | **104.10** | 2.02 |
| pr264 | 264 | **4701.00** | 4975.00 | 4834.00 | 1.26 | **4701.00** | 5100.00 | 4859.00 | 1.01 | **4701.00** | **4951.00** | **4729.60** | 1.69 |
| a280 | 280 | 114.00 | 120.00 | 117.20 | 1.94 | 116.00 | 122.00 | 118.90 | 1.13 | **113.00** | **119.00** | **115.70** | 2.14 |
| pr299 | 299 | 2259.00 | 2441.00 | 2353.10 | 1.66 | 2026.00 | 2401.00 | 2244.70 | 2.03 | **1140.00** | **2080.00** | **1503.10** | 3.74 |
| lin318 | 318 | 1774.00 | 1896.00 | 1850.90 | 2.06 | 1731.00 | 1902.00 | 1817.80 | 1.82 | **1331.00** | **1632.00** | **1368.40** | 3.81 |
| rd400 | 400 | 544.00 | 565.00 | 554.30 | 3.04 | 533.00 | 562.00 | 550.80 | 2.55 | **467.00** | **559.00** | **528.20** | 3.86 |
| fl417 | 417 | 1159.00 | 1206.00 | 1180.90 | 2.80 | 1082.00 | 1182.00 | 1151.10 | 2.37 | **999.00** | **1100.00** | **1045.30** | 4.81 |
| gr431 | 431 | 5787.00 | 7508.00 | 6858.60 | 3.70 | 5800.00 | 7298.00 | 6181.00 | 3.22 | **4027.00** | **5100.00** | **4174.50** | 5.99 |
| pr439 | 439 | 4642.00 | 5244.00 | 4926.80 | 3.63 | 4555.00 | 5268.00 | 4828.40 | 3.21 | **2384.00** | **3094.00** | **2735.50** | 6.13 |
| pcb442 | 442 | 1803.00 | 1910.00 | 1845.60 | 3.12 | 1811.00 | 1903.00 | 1838.50 | 2.87 | **1628.00** | **1838.00** | **1179.20** | 4.75 |
| d493 | 493 | **2008.00** | **2008.00** | **2008.00** | 3.70 | **2008.00** | 2254.00 | 2055.70 | 3.19 | **2008.00** | **2008.00** | **2008.00** | 4.82 |
| att532 | 532 | 1027.00 | 1088.00 | 1045.70 | 5.34 | 1009.00 | 1060.00 | 1035.00 | 4.46 | **742.00** | **1008.00** | **890.50** | 8.33 |
| ali535 | 535 | 8063.00 | 8705.00 | 8416.50 | 6.16 | 7742.00 | 8540.00 | 8165.20 | 3.92 | **3889.00** | **4507.00** | **4209.30** | 12.78 |
| si535 | 535 | 292.00 | 305.00 | 299.60 | 4.01 | 292.00 | **302.00** | 297.10 | 4.25 | **277.00** | 305.00 | **293.00** | 6.48 |
| pa561 | 561 | 72.00 | 75.00 | 73.70 | 4.76 | 71.00 | 75.00 | 72.60 | 5.94 | **70.00** | **73.00** | **71.90** | 11.33 |
| u574 | 574 | 1151.00 | 1196.00 | 1177.30 | 5.63 | 1170.00 | 1214.00 | 1194.90 | 4.51 | **949.00** | **1189.00** | **1083.70** | 10.26 |
| rat575 | 575 | 187.00 | 195.00 | 191.70 | 5.89 | **186.00** | 196.00 | **190.10** | 6.22 | 188.00 | **194.00** | 190.20 | 8.74 |
| p654 | 654 | 3090.00 | 3195.00 | 3133.50 | 6.61 | 2925.00 | **3180.00** | 3106.50 | 6.43 | **2745.00** | 3195.00 | **2973.00** | 15.50 |
| d657 | 657 | 1466.00 | 1700.00 | 1600.10 | 8.98 | 1374.00 | 1715.00 | 1512.50 | 6.93 | **1368.00** | **1431.00** | **1380.40** | 16.64 |
| gr666 | 666 | 8429.00 | 9004.00 | 8830.80 | 8.69 | 9123.00 | 8932.00 | 8647.30 | 7.15 | **4469.00** | **4914.00** | **4531.30** | 20.51 |
| u724 | 724 | 1169.00 | 1221.00 | 1195.70 | 7.89 | 1145.00 | **1213.00** | 1188.50 | 7.18 | **1004.00** | 1220.00 | **1133.00** | 17.20 |
| rat783 | 783 | **216.00** | 228.00 | 222.40 | 9.87 | 219.00 | 227.00 | 221.70 | 8.94 | 217.00 | **224.00** | **220.00** | 18.73 |
| nrw1379 | 1379 | 1035.00 | 1071.00 | 1059.40 | 32.07 | 1045.00 | **1062.00** | 1053.20 | 35.87 | **1028.00** | 1070.00 | **1049.10** | 66.73 |
| fl1577 | 1577 | 970.00 | 988.00 | 978.70 | 49.97 | 964.00 | **986.00** | 978.80 | 42.47 | **958.00** | 988.00 | **972.40** | 98.71 |
| d1655 | 1655 | 1512.00 | 1703.00 | 1587.60 | 72.66 | 1511.00 | 1621.00 | 1562.30 | 57.66 | **1476.00** | **1568.00** | **1518.50** | 121.71 |
| vm1748 | 1748 | 8936.00 | 9288.00 | 9185.50 | 64.27 | 8996.00 | 9237.00 | 9138.90 | 59.71 | **6820.00** | **7258.00** | **7000.30** | 130.57 |
| u1817 | 1817 | **1159.00** | **1194.00** | 1182.80 | 62.75 | 1177.00 | **1194.00** | 1185.90 | 56.83 | 1163.00 | 1198.00 | **1173.40** | 105.49 |
| rl1889 | 1889 | 7883.00 | 8064.00 | 7997.40 | 68.41 | 7885.00 | 8140.00 | 8024.40 | 75.94 | **6797.00** | **7868.00** | **7422.80** | 170.28 |
| **Overall** | | 2095.04 | 2338.50 | 2221.35 | 5.77 | 2017.68 | 2303.61 | 2144.84 | 5.33 | **1591.55** | **1814.83** | **1672.98** | 11.17 |
| **NBV** | | 19 | 11 | 8 | | 25 | 20 | 13 | | **79** | **74** | **80** | |



(A) BTSP Solution for **dantzig42**



(B) BTSP Solution for **eil51**

FIGURE 4.6: Plots showing the MS-ILS algorithm solutions on various TSPLIB instances

The performance of the three MS-ILS variants, MS-ILS($h_1$), MS-ILS($h_2$), and MS-ILS($h_1 + h_2$) is shown in Table 4.2. The first column in this table reflects the instance's name. The number of nodes for each instance is reported in the second column ($n$). The columns ($Best, Worst, Average$) give the best, worst and average maximum edge costs over 10 separate runs for each of the three variations of the proposed MS-ILS. The best results are highlighted in bold to make them stand out. The average of the execution times of 10 distinct runs are reported in the column ($Time$). The number of occasions on which the appropriate MS-ILS variant obtains a best value is reported in the last row, labelled '$NBV$'.

The MS-ILS($h_1 + h_2$) received the best values for the best and average objective function values in 79 and 80 instances respectively, out of 82. For the best and average objective function values, the MS-ILS($h_2$) obtained the best results in 25 and 13 instances respectively. For the best and average objective function values, the MS-ILS($h_1$) obtained the best results in 19 and 11 instances respectively.

The performance of MS-ILS($h_2$) is better when compared to that of MS-ILS($h_1$), as seen from Table 4.3, whereas, MS-ILS($h_1 + h_2$) performs very well.

Figure 4.6 plots the solutions obtained by our MS-ILS($h_1 + h_2$) approach for two TSPLIB instances, viz. **dantzig42** and **eil51**.

TABLE 4.3: WILCOXON SIGNED RANK TEST RESULT

| | MS-ILS($h_1+h_2$) | | | | | | | MS-ILS($h_2$) | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | *NWT/Total* | $W^+$ | $W^-$ | $Z$ | $Z_c$ | *Significant* | | *NWT/Total* | $W^+$ | $W^-$ | $Z$ | $Z_c$ | *Significant* |
| **MS-ILS**($h_2$) | 71/82 | 2553 | 3 | -7.306 | -2.576 | Yes | **MS-ILS**($h_1$) | 74/82 | 2355 | 0 | -5.212 | -2.576 | Yes |
| **MS-ILS**($h_1$) | 74/82 | 2775 | 0 | -7.475 | -2.576 | Yes | | | | | | | |

### 4.3.1 Wilcoxon signed rank test

Two tailed Wilcoxon signed rank test [25] has been deployed to check whether the performances of the three MS-ILS variants differ significantly. The significance criteria were set to 1% (i.e., $p - value \leq 0.01$). This test grades the difference between the normalised values of '*Average*' produced by our method. Table 4.3 shows the results of this statistical test. The '*NWT/Total*' in this table refers to the number of TSPLIB instances without a tie as a percentage of the total number of cases compared. $W^+$ represents the sum of ranks for cases where the top of the table approach (MS-ILS($h_1 + h_2$)/MS-ILS($h_2$)) outperforms its competitor on the left side of the table, whereas $W^-$ represents the sum of ranks for cases where the top of the table approach (MS-ILS($h_1 + h_2$)/MS-ILS($h_2$)) under performs its competitor on the left side of the table. Since there are more than thirty instances ($NWT > 30$), the test statistic $Z$ is utilized. According to the Wilcoxon signed rank test, the $Z$ value is compared to the critical value $Z_c$. If $Z \leq Z_c$, the performance of the two MS-ILS variations under consideration differs significantly; otherwise, the difference is insignificant. The results of all three MS-ILS variants (MS-ILS($h_1$), MS-ILS($h_2$), and MS-ILS($h_1 + h_2$) are statistically significant from each other, as shown in Table 4.3.

## 4.4 Time Complexity Analysis

The average time complexity of the proposed MS-ILS algorithm for solving the BTSP depends mainly on the two heuristics proposed in 3.2.3. In this section, we attempt to analyze the average time complexity of the two heuristics separately and validate them with experimental results.

### 4.4.1 Average Time Complexity of Heuristic $h_1$

A tour on k nodes is denoted by $H^k$=$\{c_1, c_2, c_3, \ldots, c_k\}$. Choose a random node $r$ from the remaining $n-k$ nodes and place it between the nodes with the highest edge cost in $H^k$. On $k+1$ nodes, replace $H^k$ with the new tour $H^{k+1}$, and repeat until a tour $H^n$ is achieved. This method takes $O(n^3)$ time to implement in a simple way. The following method can decrease the time complexity to $O(n^2)$. Let

$$Max(H^k) = \max_{1 \le i \le k} \{l_{(c_i, c_{i+1})}\} = l_{(c_p, c_{p+1})} \tag{4.1}$$

and

$$Max(H^k - \{(c_p, c_{p+1})\}) = \max_{1 \le i \le k} \{l_{(c_i, c_{i+1})}\}, \qquad i \neq p \tag{4.2}$$

where $k+1 \equiv 1$. Because the index p in (4.1) is not guaranteed to be unique, $Max(H^k)$ might be equivalent to $Max(H^k - (c_p, c_{p+1}))$ and hence, the cost $\Delta_i^r$ of inserting node between nodes $c_i$ and $c_{i+1}$ is given by

$$\Delta_i^r = \begin{cases} max\{Max(H^k), l_{(c_i, c_r)}, l_{(c_r, c_{i+1})}\}, & \text{if } i \neq p \\ max\{Max(H^k - \{(c_p, c_{p+1})\}), l_{(c_i, c_r)}, l_{(c_r, c_{i+1})}\}, & \text{if } i = p \end{cases} \tag{4.3}$$

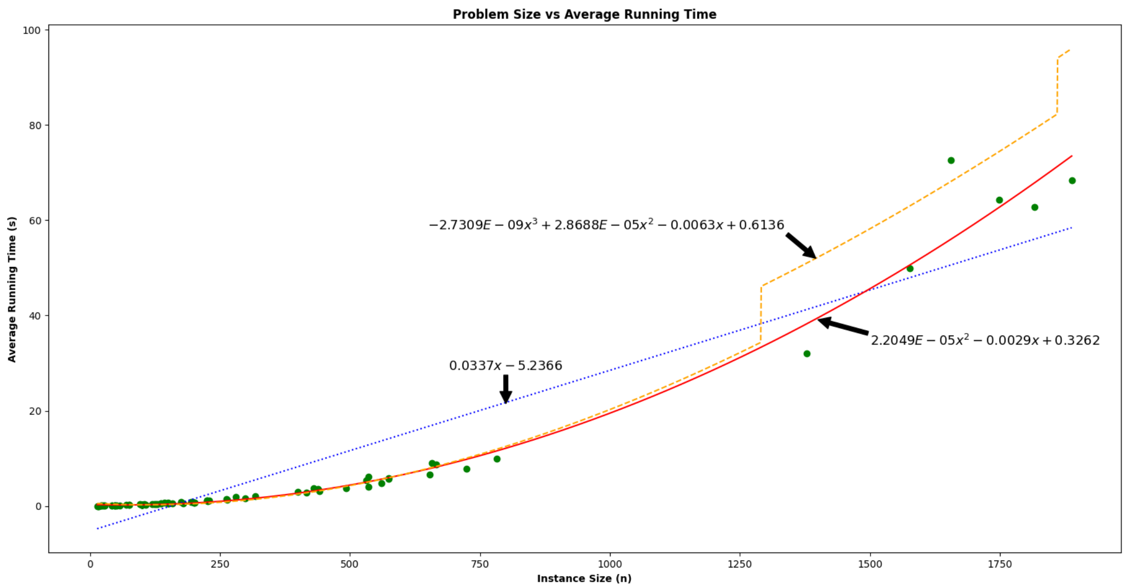



FIGURE 4.7: Problem Size vs Average Running Time for Heuristic $h_1$

Choose $q$ such that $\Delta_q^r = \min_{1\leq i\leq k} \Delta_i^r$. The new cycle $H^{k+1}$ on $k+1$ nodes is then obtained by inserting node $r$ between nodes $c_q$ and $c_{q+1}$ in $H^k$. The heuristic generates a $H^n$ tour on $n$ nodes. Given the values $Max(H^k)$ and $Max(H^k - \{c_p, c_{p+1}\})$ for $H^k$ and the index $q$, the corresponding values for $H^{k+1}$ may be found in $O(1)$ time. It should be emphasised that updating other pertinent information every iteration requires $O(n)$ time, including computation of $q$. The complexity of the heuristic $h_1$ may thus be validated as $O(n^2)$.

Figure 4.7 shows that the average time complexity from the computational results obtained is also $O(n^2)$.

### 4.4.2 Average Time Complexity of Heuristic $h_2$

In each 2-opt move, two edges $(p_1, p_2)$ and $(q_1, q_2)$ are deleted where $p_1, p_2, q_1, q_2$ are all distinct, thus creating two sub-tours which are reconnected with edges $(p_1, q_1)$ and $(p_2, q_2)$ in case the replacement reduces the maximum edge length in the tour.

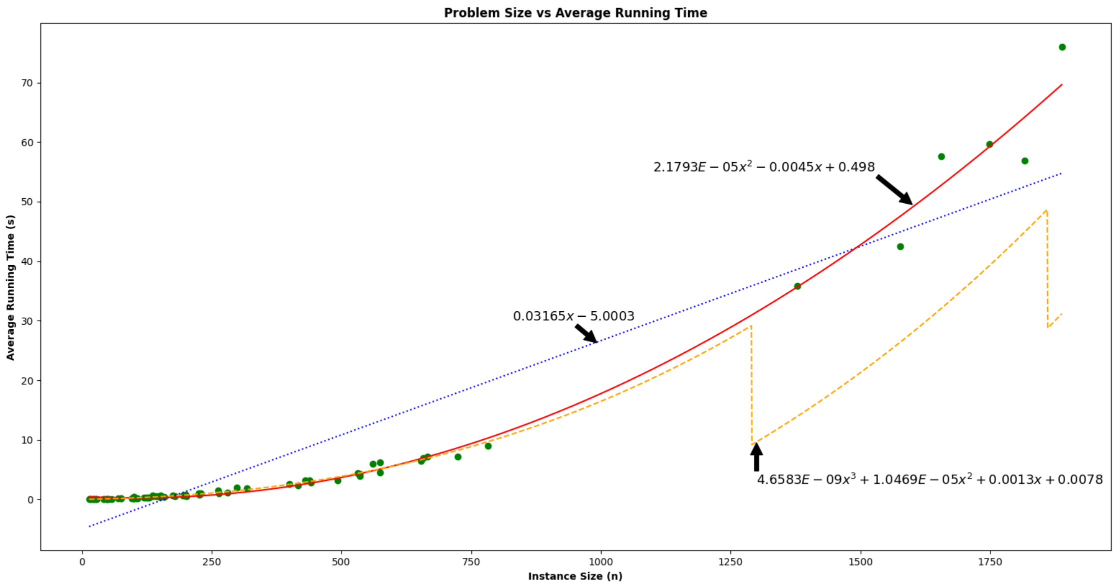



FIGURE 4.8: Problem Size vs Average Running Time for Heuristic $h_2$

In general, two-opt move incurs $O(n^2)$ cost in the worst case as we need to choose two edges from $n$ edges which is $\binom{n}{2} = n(n-1) \approx O(n^2)$. In our algorithm, since we choose the one of the edges to be the maximum edge in the tour, we need to select one edge from the remaining $(n-1)$ edges, which is $O(n)$. After the two-opt move, updating the edge with maximum edge cost is $O(n)$. Therefore, the average time complexity of the two-opt move per iteration is $O(n)$. Since the two-opt move is done n-times for n edges, the overall average time complexity becomes $n * O(n) = O(n^2)$.

Figure 4.8 shows that the average time complexity from the computational results obtained is also $O(n^2)$.

A linear, quadratic, and cubic fit has been done to estimate the running time of our approach as a function of the instance size using the enormous quantity of data obtained in the tests. Figure 4.9 shows how closely the quadratic fit approximates the running time. This is consistent with the known experimental findings for the LK-heuristic [26–30] on the average complexity of $O(n^{2.2})$.

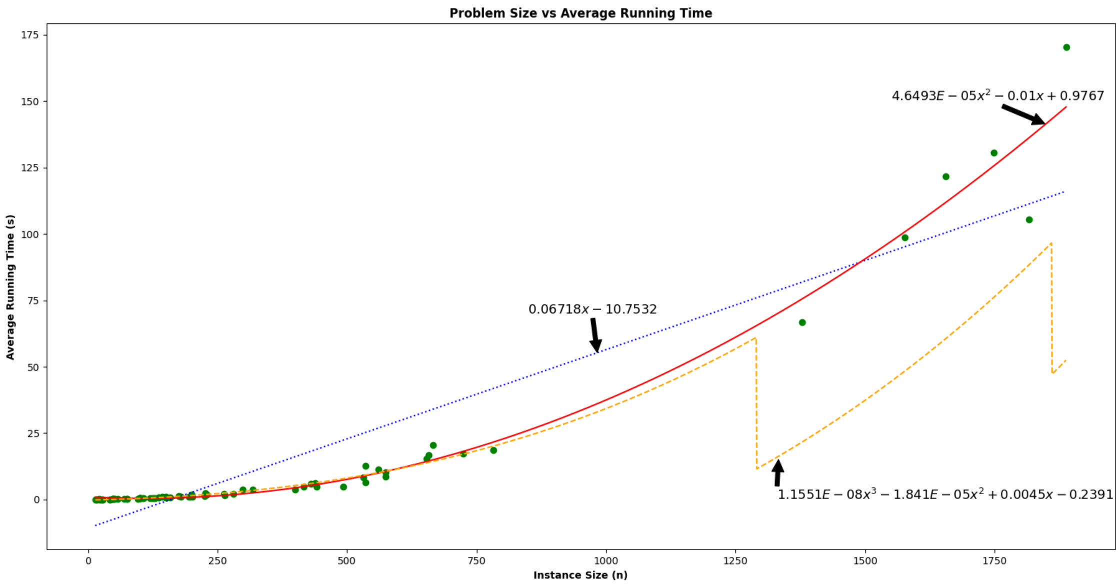



FIGURE 4.9: Problem Size vs Average Running Time for the proposed MS-ILS algorithm

# Chapter 5

# Conclusion and Future Works

## 5.1 Conclusion

To solve the bottleneck travelling salesman problem (BTSP), a multi-iterated iterated local search method has been presented. The suggested insertion and modified 2-opt move based local searches are the key components of this approach. On standard TSPLIB instances, the proposed heuristic method (viz. MS-ILS($h_1 + h_2$)) performed well when compared with two other heuristics (viz. MS-ILS($h_1$) and MS-ILS($h_2$)). MS-ILS($h_2$) performed significantly better than MS-ILS($h_1$). When compared to other existing algorithms, MS-ILS($h_1 + h_2$) proved to be the best in terms of both solution and computational time. The proposed MS-ILS technique is the first meta-heuristic approach to solve the BTSP and hence will serve as a benchmark for any future meta-heuristic approaches. Other similar problems, such as the vehicle routing problem (VRP), maximum travelling salesman problem (MTSP) can be solved using this approach.

As a future work, we intend to build a population-based meta-heuristic method for the BTSP by combining it with MS-ILS components to enhance the result.

# Bibliography


[1] P. C. Gilmore and R. E. Gomory, “Sequencing a one state-variable machine: A solvable case of the traveling salesman problem,” *Operations research*, vol. 12, no. 5, pp. 655–679, 1964.

[2] S. N. Kabadi, “Polynomially solvable cases of the tsp,” in *The traveling salesman problem and its variations*, pp. 489–583, Springer, 2007.

[3] S. N. Kabadi and A. P. Punnen, “The bottleneck tsp,” in *The traveling salesman problem and its variations*, pp. 697–736, Springer, 2007.

[4] P. Kljaus, “A special case of the bottleneck traveling salesman problem. vesci akad. navuk bssr, ser. fiz,” *Mat. Navuk*, vol. 141, no. 1, pp. 61–65, 1975.

[5] G. S. Manku, “A linear time algorithm for the bottleneck biconnected spanning subgraph problem,” *Information processing letters*, vol. 59, no. 1, pp. 1–7, 1996.

[6] G. L. Vairaktarakis, “On gilmore–gomory’s open question for the bottleneck tsp,” *Operations Research Letters*, vol. 31, no. 6, pp. 483–491, 2003.

[7] J. A. van der Veen, “An o (n) algorithm to solve the bottleneck traveling salesman problem restricted to ordered product matrices,” *Discrete applied mathematics*, vol. 47, no. 1, pp. 57–75, 1993.

[8] E. Gabovic, A. Ciz, and A. Jalas, “The bottleneck traveling salesman problem,” *Trudy Vy cisl. Centra Tartu. Gos. Univ*, vol. 22, pp. 3–24, 1971.

[9] M.-Y. Kao and M. Sanghi, “An approximation algorithm for a bottleneck traveling salesman problem,” *Journal of Discrete Algorithms*, vol. 7, no. 3, pp. 315–326, 2009.

[10] N. Doroshko and V. Sarvanov, "The minimax traveling salesman problem and hamiltonian cycles in powers of graphs.(russian) vestsi akad. navuk bssr ser. fi z," *Mat. Navuk*, vol. 143, no. 6, pp. 119–120, 1981.

[11] R. G. Parker and R. L. Rardin, "Guaranteed performance heuristics for the bottleneck travelling salesman problem," *Operations Research Letters*, vol. 2, no. 6, pp. 269–272, 1984.

[12] V. Sarvanov, "Traveller minimax problem in plane: Complexity of approximate solution," in *DOKLADY AKADEMII NAUK BELARUSI*, vol. 39, pp. 16–19, ACADEMII NAUK BELARUSI F SCORINA PR 66, ROOM 403, MINSK 220072, BYELARUS, 1995.

[13] R. S. Garfinkel and K. Gilbert, "The bottleneck traveling salesman problem: Algorithms and probabilistic analysis," *Journal of the ACM (JACM)*, vol. 25, no. 3, pp. 435–448, 1978.

[14] R. Ramakrishnan, P. Sharma, and A. P. Punnen, "An efficient heuristic algorithm for the bottleneck traveling salesman problem," *Opsearch*, vol. 46, no. 3, pp. 275–288, 2009.

[15] Z. H. Ahmed, "A hybrid sequential constructive sampling algorithm for the bottleneck traveling salesman problem," *International Journal of Computational Intelligence Research*, vol. 6, no. 3, pp. 475–484, 2010.

[16] Z. H. Ahmed, *A sequential constructive sampling and related approaches to combinatorial optimization.* PhD thesis, Tezpur University, Assam, India, 2000.

[17] Z. H. Ahmed, "A hybrid genetic algorithm for the bottleneck traveling salesman problem," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 12, no. 1, pp. 1–10, 2013.

[18] H. R. Lourenço, O. C. Martin, and T. Stützle, "Iterated local search," in *Handbook of metaheuristics*, pp. 320–353, Springer, 2003.

[19] H. R. Lourenço, O. C. Martin, and T. Stützle, "Iterated local search: Framework and applications," in *Handbook of metaheuristics*, pp. 129–168, Springer, 2019.

[20] A. Subramanian and M. Battarra, "An iterated local search algorithm for the travelling salesman problem with pickups and deliveries," *Journal of the Operational Research Society*, vol. 64, no. 3, pp. 402–409, 2013.

[21] P. H. V. Penna, A. Subramanian, and L. S. Ochi, "An iterated local search heuristic for the heterogeneous fleet vehicle routing problem," *Journal of Heuristics*, vol. 19, no. 2, pp. 201–232, 2013.

[22] M. M. Silva, A. Subramanian, and L. S. Ochi, "An iterated local search heuristic for the split delivery vehicle routing problem," *Computers & Operations Research*, vol. 53, pp. 234–249, 2015.

[23] D. P. Cuervo, P. Goos, K. Sörensen, and E. Arráiz, "An iterated local search algorithm for the vehicle routing problem with backhauls," *European Journal of Operational Research*, vol. 237, no. 2, pp. 454–464, 2014.

[24] P. Venkatesh, G. Srivastava, and A. Singh, "A multi-start iterated local search algorithm with variable degree of perturbation for the covering salesman problem," in *Harmony Search and Nature Inspired Optimization Algorithms*, pp. 279–292, Springer, 2019.

[25] F. Wilcoxon, S. Katti, and R. A. Wilcox, *Critical values and probability levels for the Wilcoxon rank sum test and the Wilcoxon signed rank test.* American Cyanamid Company, 1963.

[26] K. Helsgaun, "An effective implementation of the lin–kernighan traveling salesman heuristic," *European journal of operational research*, vol. 126, no. 1, pp. 106–130, 2000.

[27] D. S. Hochbaum and D. B. Shmoys, "A unified approach to approximation algorithms for bottleneck problems," *Journal of the ACM (JACM)*, vol. 33, no. 3, pp. 533–550, 1986.

[28] D. Johnson and L. McGeoch, "Benchmark code and instance generation codes," *Last Updated May*, 2002.

[29] D. S. Johnson, G. Gutin, L. A. McGeoch, A. Yeo, W. Zhang, and A. Zverovitch, "Experimental analysis of heuristics for the atsp," in *The traveling salesman problem and its variations*, pp. 445–487, Springer, 2007.

[30] D. S. Johnson and L. A. McGeoch, "Experimental analysis of heuristics for the stsp," in *The traveling salesman problem and its variations*, pp. 369–443, Springer, 2007.

[31] P. Venkatesh, A. Singh, and R. Mallipeddi, "A multi-start iterated local search algorithm for the maximum scatter traveling salesman problem," in *2019 IEEE Congress on Evolutionary Computation (CEC)*, pp. 1390–1397, 2019.