Multiple Start and Iterated Local Search for the Hamiltonian Cycle Problem

Evolutionary Computation Laboratory: Assignment 6

151927 Samuel Janas, samuel.janas@student.put.poznan.pl
151955 Bruno Urbaniak, bruno.urbaniak@student.put.poznan.pl
151960 Patryk Maciejewski, patryk.maciejewski.1@student.put.poznan.pl

1 Introduction

In this report, we extend our exploration of local search methods for the Hamiltonian Cycle Problem (HCP) by implementing and evaluating two advanced techniques: **Multiple Start Local Search ()** and **Iterated Local Search (ILS)**. These approaches aim to improve upon the basic steepest local search algorithm by introducing multiple starting solutions and a perturbation mechanism to escape local optima.

The goal is to assess whether these extensions can achieve better results than the previously implemented methods, including steepest local search and candidate moves. The computational experiments involve evaluating MSLS and ILS on the same datasets, ensuring comparability of results.

2 Problem Description and Implementation

2.1 Hamiltonian Cycle Problem Overview

The HCP involves finding a closed loop that visits a subset of nodes exactly once. In our study, the objective is to select 50% of the available nodes to form such a cycle while minimizing the combined objective function, comprising the total cost of selected nodes and the total distance of the cycle.

2.2 Implemented Methods

2.2.1 Multiple Start Local Search (MSLS)

The MSLS method involves running the steepest local search algorithm multiple times, each starting from a random initial solution. The best solution found across all runs is selected as the final solution. This approach mitigates the risk of being trapped in local optima by diversifying the starting points.

- **Initialization**: Generate 200 random starting solutions.
- **Local Search**: Apply steepest local search with edge exchange moves for each starting solution.
- **Selection**: Record the best solution among all runs.

2.2.2 Iterated Local Search (ILS)

ILS builds upon a single local search run by iteratively perturbing the solution and applying local search again. The perturbation introduces controlled randomness to escape local optima.

- **Initialization**: Start from a random initial solution.
- **Perturbation**: Modify the current solution by permuting a fraction ($p = 30\%$) of the selected nodes.

- **Local Search**: Apply steepest local search with edge exchange moves to the perturbed solution.
- **Acceptance Criterion**: Accept the new solution if it improves upon the current best solution.
- **Termination**: Stop after a duration equivalent to the average running time of MSLS.

2.3 Pseudocode for Implemented Algorithms

Problem 1: Multiple Start Local Search (MSLS)

1. **Input**:
   - Cost matrix $costMatrix$
   - Number of iterations n (200)
2. **Output**:
   - Best solution $bestSolution$
   - Best fitness $bestFitness$
3. **Procedure**:
   (i) Initialize $bestFitness \leftarrow \infty$
   (ii) For $i = 1$ to n:
   - Generate a random solution $solution$
   - Apply steepest local search:

$$solution \leftarrow \text{SteepestLocalSearch}(solution, costMatrix)$$

   - Evaluate fitness $fitness \leftarrow \text{Fitness}(solution, costMatrix)$
   - If $fitness < bestFitness$:
     - Update $bestSolution \leftarrow solution$
     - Update $bestFitness \leftarrow fitness$

   (iii) Return $bestSolution, bestFitness$

Problem 2: Iterated Local Search (ILS)

1. **Input**:
   - Cost matrix $costMatrix$
   - Time limit t
   - Perturbation percentage p (30%)
2. **Output**:
   - Best solution $bestSolution$
   - Best fitness $bestFitness$
3. **Procedure**:
   (i) Initialize $bestFitness \leftarrow \infty$
   (ii) Generate a random solution $solution$
   (iii) Start timer

(iv) While time elapsed $< t$:

- Perturb the solution:

$$perturbedSolution \leftarrow \text{Perturb}(solution, p)$$

- Apply steepest local search:

$$newSolution \leftarrow \text{SteepestLocalSearch}(perturbedSolution, costMatrix)$$

- Evaluate fitness $fitness \leftarrow \text{Fitness}(newSolution, costMatrix)$
- If $fitness < bestFitness$:
  - Update $bestSolution \leftarrow newSolution$
  - Update $bestFitness \leftarrow fitness$

(v) Return $bestSolution, bestFitness$

Problem 3: Perturbation in ILS

1. **Input**:
   - Current solution *solution*
   - Perturbation percentage p
2. **Output**:
   - Perturbed solution *perturbedSolution*
3. **Procedure**:
   (i) Compute the number of nodes to perturb:

$$k \leftarrow \lfloor p \times \text{length}(solution) \rfloor$$

   (ii) Select a random starting index *start*
   (iii) For each index i in the range $[start, start + k)$:
   - Replace node $solution[i \mod \text{length}(solution)]$ with a randomly generated node not in the current solution.

   (iv) Return *perturbedSolution*

3 Computational Experiment

3.1 Experiment Setup

- **Methods Compared:**
  - Multiple Start Local Search (MSLS) with 200 iterations of steepest local search.
  - Iterated Local Search (ILS) with a stopping criterion based on the average runtime of MSLS.
- **Instances Used:**
  - Instance A (TSPA)
  - Instance B (TSPB)
- **Metrics Recorded:**

- Best, worst, and average objective function values.
- Execution times for each run of MSLS and ILS.
- Number of local search runs conducted within ILS.

3.2 Execution Time Analysis for MSLS

Instance A (TSPA): The execution times for 20 runs of MSLS on TSPA were recorded as follows:

Execution Times (seconds): [21.05, 23.67, 24.38, 24.67, 24.59, 24.25, 24.67, 24.98, 24.86, 25.32, 25.11, 24.72, 25.11, 24.97, 25.91, 25.26, 25.16, 24.90, 24.59, 24.83].

The average runtime for TSPA was calculated as:

$$\text{Average Runtime (TSPA)} = \frac{\sum \text{Execution Times}}{20} \approx 24.75 \text{ seconds.}$$

Instance B (TSPB): Similarly, the execution times for 20 runs of MSLS on TSPB were recorded as follows:

Execution Times (seconds): [22.96, 24.69, 24.67, 25.03, 24.73, 24.61, 24.77, 25.11, 25.06, 24.72, 24.87, 24.73, 25.83, 25.16, 24.91, 25.24, 24.75, 24.73, 24.90, 24.59].

The average runtime for TSPB was calculated as:

$$\text{Average Runtime (TSPB)} = \frac{\sum \text{Execution Times}}{20} \approx 24.84 \text{ seconds.}$$

3.3 ILS Execution Setup

The stopping condition for ILS was set to the average runtime of MSLS for each instance. Specifically:

- For TSPA, the ILS algorithm was executed with a time limit of **24.75 seconds**.
- For TSPB, the ILS algorithm was executed with a time limit of **24.84 seconds**.

During this period, ILS applied multiple iterations of steepest local search, interspersed with perturbations to escape local optima.

3.3.1 Number of Local Search Runs in ILS

ILS Iteration	TSPA LS Runs	TSPB LS Runs
1	572	598
2	558	566
3	567	575
4	590	553
5	572	573
6	579	579
7	582	568
8	570	564
9	574	561
10	572	561
11	572	570
12	565	573
13	573	575
14	560	555
15	567	554
16	558	548
17	563	558
18	572	542
19	577	551
20	577	566

Table 1: Number of Local Search Runs for ILS across 20 iterations for TSPA and TSPB instances.

This setup ensures that ILS and MSLS are evaluated within comparable computational budgets, enabling a fair comparison of their performance.

3.4 Results

3.4.1 Objective Function Values

Table 2: Objective Function Values for All Methods and Instances, Sorted by Average Fitness

Method	**Instance A (TSPA)**	**Instance B (TSPB)**
Random Solution	264724.25 (225047 – 290346)	213180.80 (190200 – 235839)
Nearest Neighbor (End Insertion)	85108.51 (83182 – 89433)	54390.43 (52319 – 59030)
Nearest Neighbor (Flexible Insertion)	73178.55 (71179 – 75450)	45870.25 (44417 – 53438)
Greedy Cycle	72634.87 (71488 – 74410)	51397.91 (49001 – 57262)
Greedy Heuristic (Weighted Sum)	73762.84 (71544 – 76341)	50992.64 (46990 – 58454)
Greedy 2-Regret Heuristic	73731.69 (71809 – 76323)	50794.27 (45814 – 59121)
LS Random Steepest Intranode	88159.52 (80186 – 96426)	63017.28 (55773 – 70444)
LS Random Greedy Intranode	86187.72 (79078 – 93575)	61026.47 (54087 – 69709)
LS Random Steepest Intraedge	73863.17 (71355 – 79486)	48364.52 (45452 – 51331)
LS Random Greedy Intraedge	73836.13 (71571 – 77616)	48330.82 (45905 – 52361)
LS Nearest Neighbour Flexible Greedy Intranode	72785.85 (71034 – 74904)	45450.70 (43826 – 50886)
LS Nearest Neighbour Flexible Greedy Intraedge	71173.25 (69997 – 73545)	45021.37 (43790 – 50495)
LS Nearest Neighbour Flexible Steepest Intranode	72805.67 (71034 – 74904)	45414.50 (43826 – 50876)
LS Nearest Neighbour Flexible Steepest Intraedge	70972.69 (69864 – 73068)	44976.43 (43921 – 50495)
Steepest LS with Candidate Moves (k = 10)	74433.93 (71729 – 78375)	48741.32 (46322 – 52626)
Steepest LS with Move Evaluations	74056.84 (70801 – 78688)	48425.14 (45605 – 51662)
MSLS	71326.7 (70802 – 71851)	45798.3 (45207 – 46236)
ILS	70386.5 (69836 – 70891)	45048.3 (44449 – 45803)

4 2D Visualization of the Best Solutions

4.1 Instance A Best Solution Visualizations

4.1.1 MSLS for Instance A

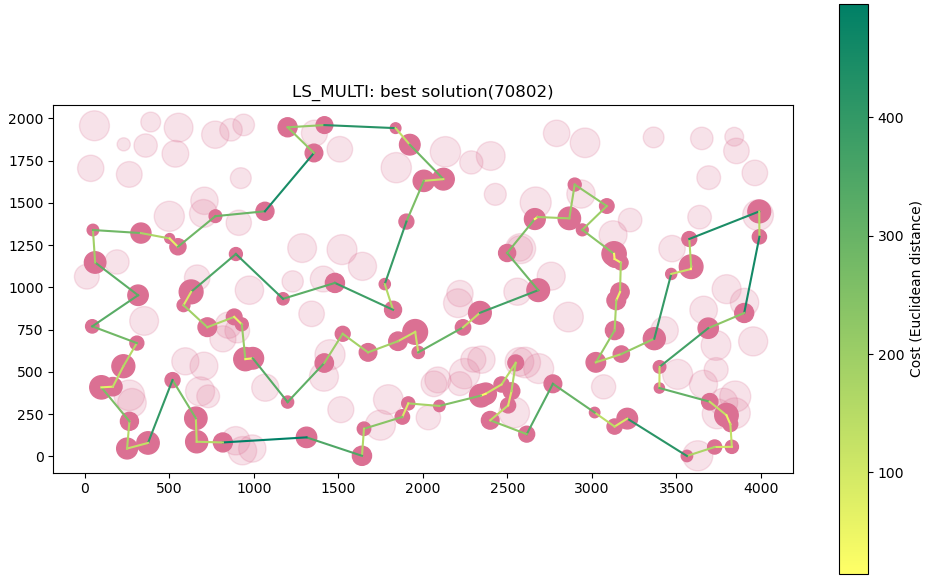

Figure 1: Best solution for Instance A (TSPA) using MSLS.

4.1.2 ILS for Instance A

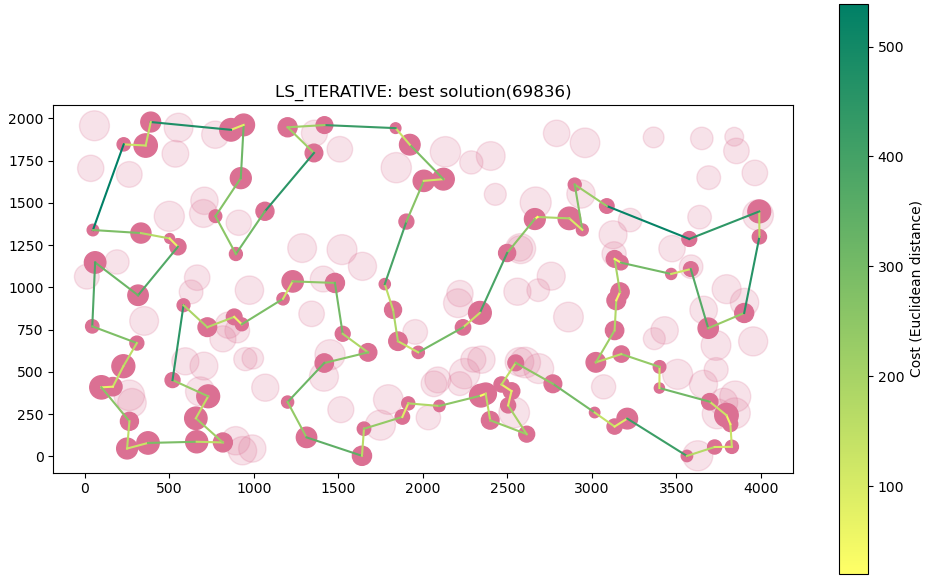

Figure 2: Best solution for Instance A (TSPA) using ILS.

4.2 Instance B Best Solution Visualizations

4.2.1 MSLS for Instance B

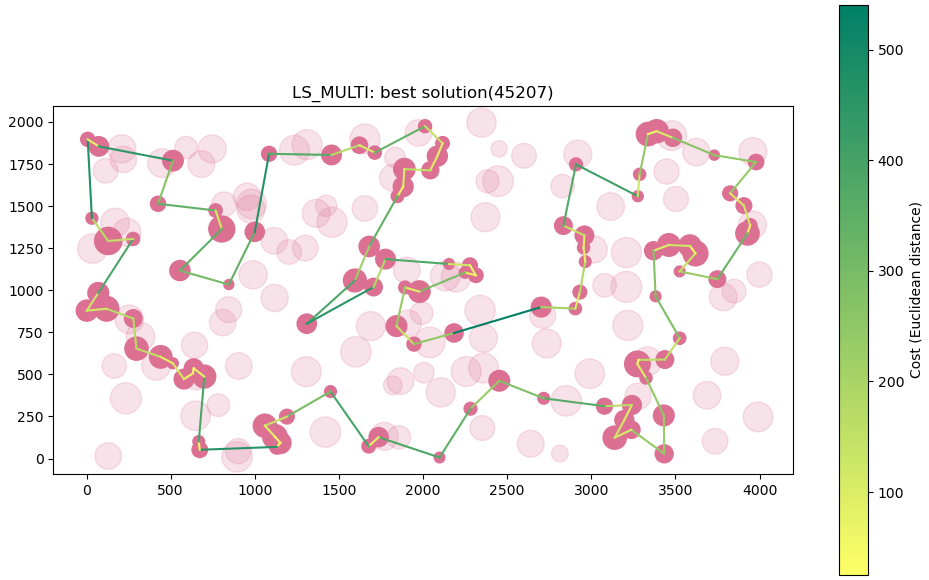

Figure 3: Best solution for Instance B (TSPB) using MSLS.

4.2.2 ILS for Instance B

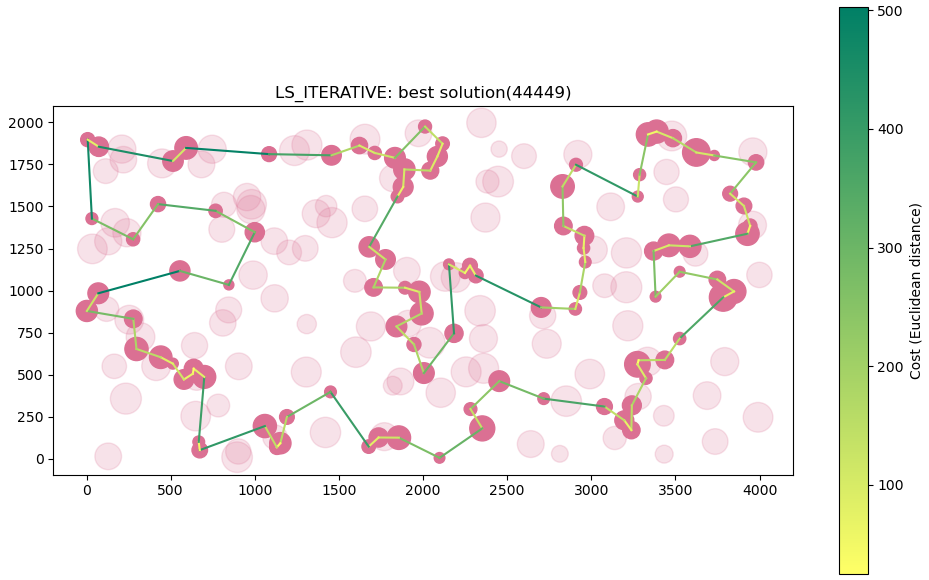

Figure 4: Best solution for Instance B (TSPB) using ILS.

4.2.3 Best Solutions for Each Method

Table 3: Best Solutions for Instances A and B by Method

Instance	Method	Best Solution
TSPA	MSLS	[79, 122, 63, 94, 124, 167, 148, 9, 62, 102, 144, 14, 49, 3, 178, 106, 52, 55, 57, 129, 92, 179, 185, 40, 119, 165, 27, 90, 81, 196, 145, 78, 31, 56, 113, 175, 171, 16, 25, 44, 120, 2, 75, 86, 101, 1, 152, 97, 26, 100, 53, 180, 154, 135, 70, 127, 112, 4, 84, 184, 190, 10, 177, 30, 54, 48, 160, 34, 181, 146, 22, 159, 193, 41, 139, 46, 0, 117, 143, 183, 89, 186, 23, 137, 176, 80, 51, 59, 115, 5, 42, 43, 116, 65, 131, 149, 123, 162, 151, 133]
TSPB	MSLS	[198, 156, 1, 16, 27, 38, 135, 102, 63, 40, 107, 133, 122, 90, 125, 131, 121, 51, 147, 6, 188, 169, 70, 3, 15, 145, 13, 195, 168, 139, 182, 25, 138, 11, 29, 109, 35, 0, 160, 33, 104, 8, 111, 143, 106, 124, 62, 18, 55, 34, 152, 183, 140, 4, 149, 28, 20, 60, 148, 47, 94, 66, 179, 185, 22, 99, 130, 95, 86, 166, 194, 176, 180, 113, 114, 137, 127, 165, 89, 103, 163, 153, 81, 77, 141, 61, 36, 177, 5, 45, 142, 78, 175, 80, 190, 73, 54, 31, 193, 117]
TSPA	ILS	[2, 120, 44, 25, 16, 171, 175, 113, 56, 31, 78, 145, 92, 129, 57, 55, 52, 178, 106, 185, 40, 196, 81, 90, 27, 165, 14, 144, 49, 102, 62, 9, 148, 124, 94, 63, 79, 80, 176, 137, 23, 186, 89, 183, 143, 117, 0, 46, 115, 139, 68, 93, 140, 108, 69, 18, 22, 159, 193, 41, 181, 146, 34, 160, 48, 54, 30, 177, 10, 190, 4, 112, 84, 35, 184, 42, 43, 116, 65, 59, 118, 51, 151, 133, 162, 123, 127, 70, 135, 154, 180, 53, 100, 26, 86, 75, 101, 1, 97, 152]
TSPB	ILS	[149, 28, 59, 20, 60, 148, 47, 94, 66, 99, 130, 95, 86, 185, 179, 172, 52, 166, 194, 176, 180, 113, 103, 127, 89, 163, 153, 81, 77, 97, 141, 91, 61, 36, 177, 5, 78, 175, 45, 80, 190, 73, 54, 31, 193, 117, 198, 156, 1, 27, 38, 131, 121, 51, 90, 122, 135, 63, 40, 107, 133, 10, 147, 6, 188, 169, 132, 70, 3, 15, 145, 13, 195, 168, 139, 11, 138, 33, 160, 144, 104, 8, 82, 111, 29, 0, 109, 35, 143, 106, 124, 62, 18, 55, 34, 170, 152, 183, 140, 4]

5 Conclusion

This study demonstrates the potential of MSLS and ILS in improving solution quality for the HCP. While MSLS provides a straightforward approach with multiple random starts, ILS leverages perturbations to achieve superior results within the same computational budget.