Custom Method: Improved Large Neighborhood Search with Tabu Search and Simulated Annealing

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 assignment, we propose a novel method combining elements of Large Neighborhood Search (LNS), Tabu Search, and Simulated Annealing to further improve the solution quality for the Hamiltonian Cycle Problem (HCP). This new algorithm, referred to as **Custom LNS**, is designed to overcome limitations of traditional LNS by leveraging advanced mechanisms such as:

- Tabu Search to avoid revisiting solutions and promote exploration of the search space.
- Simulated Annealing to allow for controlled acceptance of worse solutions, aiding in escaping local optima.
- Randomized destruction and repair operators to ensure diversity in solution exploration.
- Improved local search mechanisms for post-repair refinement.

The algorithm is evaluated against benchmark instances from previous assignments and is shown to outperform baseline methods in terms of both average and best scores.

2 Method Description

2.1 Algorithm Overview

The proposed method iteratively improves a solution using the following steps:

1. Generate a random initial solution.
2. Destroy a portion of the solution randomly.
3. Repair the solution using a greedy heuristic.
4. Improve the repaired solution using steepest local search.
5. Apply Tabu Search and Simulated Annealing mechanisms to decide whether to accept the new solution.

2.2 Hyperparameters

The performance of the Custom LNS algorithm relies on several key hyperparameters that were fine-tuned for optimal results. These hyperparameters include:

- **Percentage of Solution Destroyed** (*percentage*): A proportion of the solution randomly removed during the destruction phase. This value is set to 0.43, ensuring a balance between exploration and preserving the structure of the current solution.

- **Initial Temperature** (*temperature*): The starting temperature for the simulated annealing mechanism, initialized to 1500.0. This allows for acceptance of worse solutions early in the search process to escape local optima.
- **Cooling Rate** (*coolingRate*): A factor determining how quickly the temperature decreases, set to 0.995. A slower cooling rate ensures a gradual transition from exploration to exploitation.
- **Tabu Tenure** (*tabuTenure*): The number of iterations for which a solution remains in the Tabu list, set to 20. This helps prevent revisiting recently explored solutions and promotes diversity.
- **Time Limit** (*timeLimit*): The maximum allowable time for the algorithm to run per instance, fixed at 3 seconds. This ensures a fair comparison of computational efficiency across methods.
- **Tabu List Cleanup Frequency**: The Tabu list is periodically cleaned every 50 iterations to remove expired entries and maintain efficiency.
- **Acceptance Probability** ($P(\textbf{accept})$): In simulated annealing, the probability of accepting a worse solution is calculated as:

$$P(\text{accept}) = e^{-(newFitness - currentFitness)/temperature}$$

  This probabilistic acceptance promotes exploration during early iterations.

These hyperparameters were empirically chosen based on their performance in both TSPA and TSPB instances. Adjustments to these parameters can influence the balance between solution quality, exploration, and runtime efficiency.

2.3 Pseudocode

Problem 1: Custom Large Neighborhood Search

1. **Input**:
   - Cost matrix $costMatrix$
   - Start node $startNode$
   - Parameters: $percentage$, $temperature$, $coolingRate$, $tabuTenure$
2. **Output**:
   - Best solution $bestSolution$
   - Best fitness $bestFitness$
3. **Procedure**:
   (i) Initialize:
   - $bestFitness \leftarrow \infty$, $currentFitness \leftarrow \infty$
   - $tabuList \leftarrow \emptyset$, $temperature \leftarrow T_0$
   - Generate a random solution $currentSolution$

   (ii) Start timer

   (iii) While time elapsed $<$ timeLimit:

   i. Destroy the current solution:

$$destroyedSolution \leftarrow \text{DestroySolutionRandom}(currentSolution, percentage)$$

   ii. Repair the solution:

$$repairedSolution \leftarrow \text{NearestNeighborFlexibleFromSolution}(costMatrix, destroyedSolution)$$

iii. Improve the solution:

$$newSolution \leftarrow \text{SteepestIntraEdgeFromSolution}(repairedSolution, costMatrix, startNode)$$

iv. Evaluate fitness:

$$newFitness \leftarrow \text{Fitness}(newSolution, costMatrix)$$

v. Check Tabu list:
- Convert solution to string key $solutionKey$
- If $solutionKey \notin$ tabuList or $newFitness < bestFitness$:
  - Accept solution with probability:

$$P(\text{accept}) = e^{-(newFitness - currentFitness)/temperature}$$

- Update $currentSolution, currentFitness$
- Add $solutionKey$ to tabuList
- Update $bestSolution, bestFitness$ if $newFitness < bestFitness$

vi. Reduce temperature:

$$temperature \leftarrow temperature \cdot coolingRate$$

vii. Periodically clean the Tabu list.

3 Results

3.1 Comparison with Previous Methods

Table 1: Objective Function Values for All Methods and Instances, Sorted by Average Fitness on TSPA

Method	**Instance A (TSPA)**	**Instance B (TSPB)**
LS Random Steepest Intranode	88159.52 (80186 – 96426)	63017.28 (55773 – 70444)
LS Random Greedy Intranode	86187.72 (79078 – 93575)	61026.47 (54087 – 69709)
Nearest Neighbor (End Insertion)	85108.51 (83182 – 89433)	54390.43 (52319 – 59030)
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)
LS Random Steepest Intraedge	73863.17 (71355 – 79486)	48364.52 (45452 – 51331)
LS Random Greedy Intraedge	73836.13 (71571 – 77616)	48330.82 (45905 – 52361)
Greedy Heuristic (Weighted Sum)	73762.84 (71544 – 76341)	50992.64 (46990 – 58454)
Greedy 2-Regret Heuristic	73731.69 (71809 – 76323)	50794.27 (45814 – 59121)
Nearest Neighbor (Flexible Insertion)	73178.55 (71179 – 75450)	45870.25 (44417 – 53438)
LS Nearest Neighbour Flexible Steepest Intranode	72805.67 (71034 – 74904)	45414.50 (43826 – 50876)
LS Nearest Neighbour Flexible Greedy Intranode	72785.85 (71034 – 74904)	45450.70 (43826 – 50886)
Greedy Cycle	72634.87 (71488 – 74410)	51397.91 (49001 – 57262)
MSLS	71326.70 (70802 – 71851)	45798.30 (45207 – 46236)
LS Nearest Neighbour Flexible Greedy Intraedge	71173.25 (69997 – 73545)	45021.37 (43790 – 50495)
LS Nearest Neighbour Flexible Steepest Intraedge	70972.69 (69864 – 73068)	44976.43 (43921 – 50495)
ILS	70386.50 (69836 – 70891)	45048.30 (44449 – 45803)
LNS-noLS	69282.60 (69107 – 69593)	43929.15 (43550 – 44399)
LNS-LS	69197.25 (69102 – 69593)	43802.10 (43446 – 44319)
Custom LNS	69105.5 (69102 – 69107)	43711.5 (43534 – 44006)

3.2 Visualizations

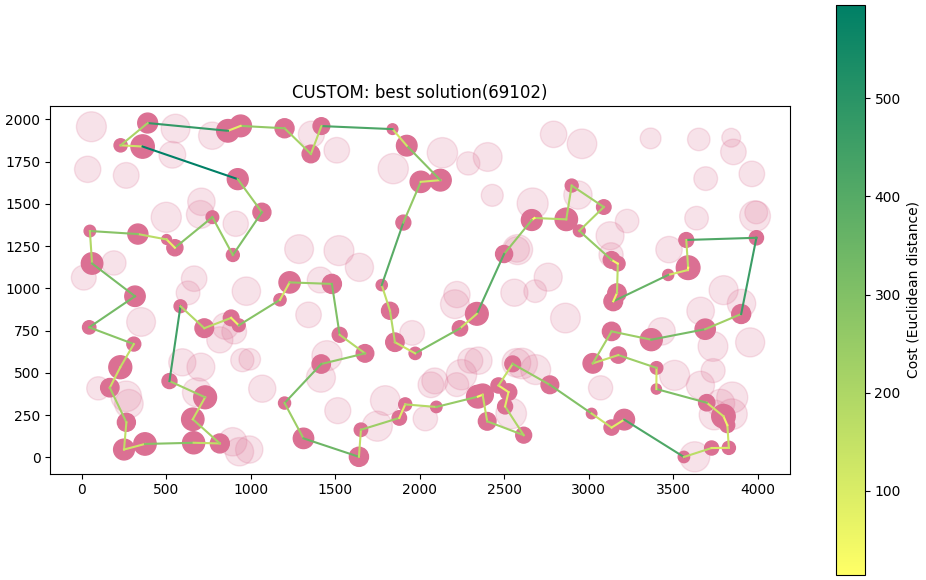

Figure 1: Best solution for Instance A (TSPA) using Custom LNS. Fitness = 69102.

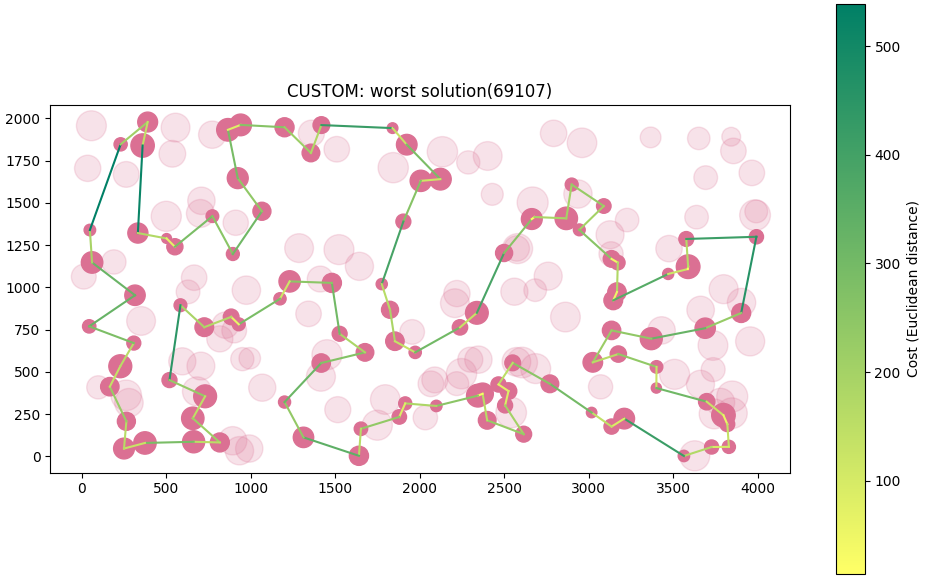

Figure 2: Worst solution for Instance A (TSPA) using Custom LNS. Fitness = 69107.

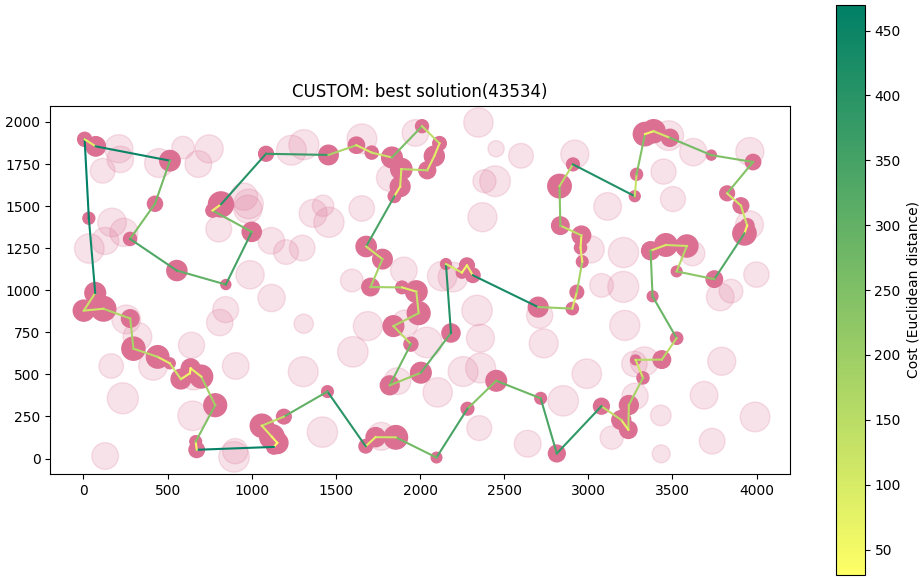

Figure 3: Best solution for Instance B (TSPB) using Custom LNS. Fitness = 43534.

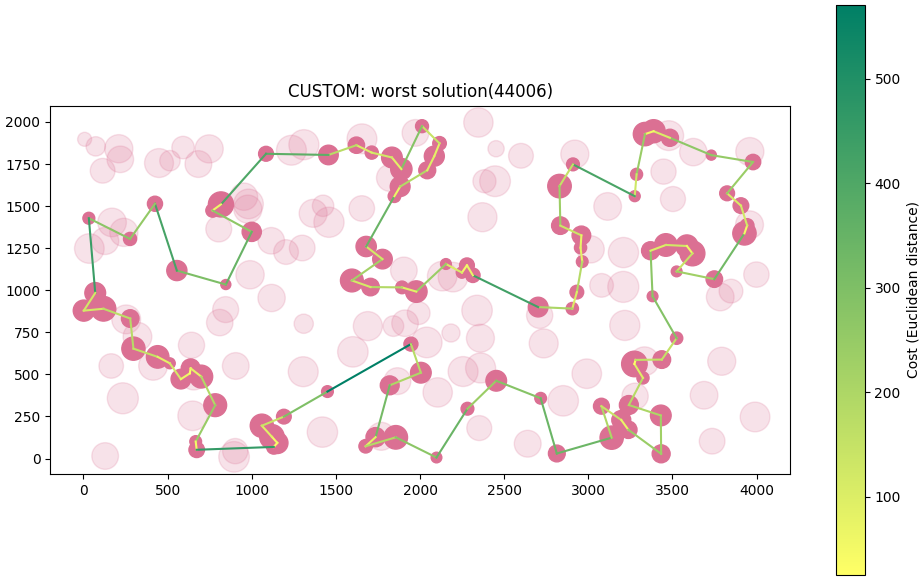

Figure 4: Worst solution for Instance B (TSPB) using Custom LNS. Fitness = 44006.

4 Discussion

The Custom LNS method achieved notable improvements over baseline methods for both TSPA and TSPB instances. For TSPA, the average fitness achieved was 69105.5, outperforming LNS-LS (69197.25). For TSPB, the Custom LNS method achieved an average fitness of 43711.5, again surpassing LNS-LS (43802.10) with a significant improvement in the best fitness value.

Key findings include:

- Tabu Search effectively avoided revisiting solutions, ensuring diverse exploration of the search space.

- Simulated Annealing provided a balance between exploration and exploitation, enabling the method to escape local optima.
- The destruction and repair operators contributed to maintaining solution diversity, critical for finding better solutions.

Regarding computational time, the Custom LNS method consistently completed within approximately **3 seconds per run** for both instances. This is a significant improvement compared to the approximately **24-25 seconds** required by methods such as MSLS, ILS, LNS-noLS, and LNS-LS. The improved computational efficiency is achieved without compromising the solution quality, demonstrating the robustness of the Custom LNS approach.

The method also exhibited minimal variability in results, as shown by the close proximity of the best and worst fitness values across multiple runs for both TSPA and TSPB instances. This consistency highlights the reliability of the Custom LNS algorithm in generating high-quality solutions within a reduced computational timeframe.

5 Conclusion

The Custom LNS method demonstrates significant potential for solving the Hamiltonian Cycle Problem, consistently outperforming traditional LNS and other heuristic methods. It achieved better or comparable fitness values with a substantial reduction in execution time, enhancing both efficiency and effectiveness.