Extension of Greedy Heuristics for Hamiltonian Cycle Problem

Evolutionary Computation Laboratory: Assignment 2

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

This report extends the previous study on heuristic methods for solving the Hamiltonian Cycle Problem (HCP). In the initial study, we implemented and analyzed several heuristic methods, including the Greedy Cycle, Nearest Neighbor (End Insertion), Nearest Neighbor (Flexible Insertion), and Random Solution methods. The purpose of this extension is to introduce and evaluate two additional heuristic methods: the **Greedy 2-Regret Heuristic** and the **Greedy Heuristic with Weighted Sum Criterion**.

The rationale behind adding these new methods is to explore alternative strategies that may provide better performance in terms of the objective function, which aims to minimize both the total path length and the total cost of selected nodes in the HCP. By incorporating the concept of regret in the selection process and combining multiple criteria with weighted sums, we aim to enhance the quality of the solutions generated and potentially outperform the previously studied methods.

2 Methodology

2.1 Greedy 2-Regret Heuristic

The Greedy 2-Regret Heuristic is an extension of the traditional greedy algorithm that incorporates the concept of regret. In this context, regret is defined as the difference in cost between the best and second-best insertion positions for a candidate node. By selecting the node with the maximum regret, the algorithm aims to make the most significant improvement at each step.

Problem 1: Greedy 2-Regret Heuristic

1. **Input**: A distance matrix distanceMatrix representing distances between nodes.
2. **Output**: A list of node indices representing a Hamiltonian cycle.
3. **Procedure**:
   - (i) Initialize the solution with a starting node.
   - (ii) Mark the node as visited.
   - (iii) While the number of nodes in the solution is less than numToSelect:
     - (a) For each unvisited node:
       - Calculate the best and second-best insertion costs into the current solution.

- Compute the regret as the difference between these two costs:

$$\text{Regret}_i = \text{SecondBestCost}_i - \text{BestCost}_i$$

(b) Select the node with the maximum regret.
(c) Insert the node at the position corresponding to the best cost.
(d) Mark the node as visited.

This approach helps in balancing local and global considerations by focusing on nodes whose exclusion would lead to the greatest increase in cost.

2.2 Greedy Heuristic with Weighted Sum Criterion

The Greedy Heuristic with Weighted Sum Criterion combines the 2-Regret heuristic with the change in the objective function, using weighted sums to evaluate candidate nodes. The algorithm assigns weights to the regret and the change in objective function, allowing for a customizable balance between these two factors.

Problem 2: Greedy Heuristic with Weighted Sum Criterion

1. **Input**: A distance matrix distanceMatrix and weights $w_{\text{regret}}, w_{\text{change}}$.
2. **Output**: A list of node indices representing a Hamiltonian cycle.
3. **Procedure**:
   (i) Initialize the solution with a starting node.
   (ii) Mark the node as visited.
   (iii) While the number of nodes in the solution is less than numToSelect:
      (a) For each unvisited node:
         - Calculate the best and second-best insertion costs into the current solution.
         - Compute the regret:

$$\text{Regret}_i = \text{SecondBestCost}_i - \text{BestCost}_i$$

         - Compute the change in the objective function:

$$\text{Change}_i = \text{BestCost}_i$$

         - Calculate the total cost using the weighted sum:

$$\text{TotalCost}_i = w_{\text{regret}} \times \text{Regret}_i + w_{\text{change}} \times \text{Change}_i$$

      (b) Select the node with the minimal total cost.
      (c) Insert the node at the position corresponding to the best cost.
      (d) Mark the node as visited.

By adjusting the weights w_{regret} and w_{change}, we can influence the algorithm's behavior to prioritize either regret or immediate cost changes.

3 Implementation Details

The new methods were implemented in Go, building upon the existing codebase used in the previous study. The key functions added are:

- `GreedyTwoRegret`: Implements the Greedy 2-Regret Heuristic.
- `GreedyRegretWeight`: Implements the Greedy Heuristic with Weighted Sum Criterion.

We utilized the following steps in the implementation:

1. Calculated the initial state using the existing `GetInitialState` function.
2. Implemented the `twoBestCandidates` function to find the two best candidate nodes based on insertion cost.
3. Calculated the regret for each candidate and selected nodes based on the algorithm's criteria.
4. For the weighted sum heuristic, set default weights ($w_{\text{regret}} = -4$, $w_{\text{change}} = 1$) based on preliminary experiments.
5. Used standard Go libraries and maintained consistency with the existing code structure to ensure compatibility.

4 Results

We conducted experiments using both new methods on the same datasets as in the previous study (Instances A and B). Each method was run for 200 iterations, and the best, worst, and average fitness values were recorded. For comparison, we also include the results of the previous methods.

4.1 Instance A Results

Method	**Best Fitness**	**Worst Fitness**	**Average Fitness**
Greedy 2-Regret Heuristic	71,809	76,323	73,731.69
Greedy Heuristic with Weighted Sum Criterion	71,544	76,341	73,762.84
Greedy Cycle	71,488	74,410	72,634.87
Nearest Neighbor (Flexible Insertion)	71,179	75,450	73,178.55
Nearest Neighbor (End Insertion)	83,182	89,433	85,108.51
Random Solution	225,047	290,346	264,724.25

Table 1: Results for Instance A showing best, worst, and average fitness values across 200 iterations for each method, including previous methods for comparison.

Method	Best Fitness	Worst Fitness	Average Fitness
Greedy 2-Regret Heuristic	45,814	59,121	50,794.27
Greedy Heuristic with Weighted Sum Criterion	46,990	58,454	50,992.64
Nearest Neighbor (Flexible Insertion)	44,417	53,438	45,870.25
Greedy Cycle	49,001	57,262	51,397.91
Nearest Neighbor (End Insertion)	52,319	59,030	54,390.43
Random Solution	190,200	235,839	213,180.80

Table 2: Results for Instance B showing best, worst, and average fitness values across 200 iterations for each method, including previous methods for comparison.

4.2 Instance B Results

5 Analysis

From the results, we observe the following:

- For **Instance A**:
  - The **Nearest Neighbor (Flexible Insertion)** method achieved the best overall fitness value (71,179), slightly outperforming all other methods.
  - The **Greedy Heuristic with Weighted Sum Criterion** and the **Greedy 2-Regret Heuristic** produced results comparable to the **Greedy Cycle** method.
  - The **Nearest Neighbor (End Insertion)** method and **Random Solution** performed significantly worse, indicating less effective exploration of the solution space.
- For **Instance B**:
  - The **Nearest Neighbor (Flexible Insertion)** method again achieved the best fitness value (44,417).
  - The **Greedy 2-Regret Heuristic** outperformed the **Greedy Cycle** method and was close to the best method.
  - The **Greedy Heuristic with Weighted Sum Criterion** showed competitive performance but did not surpass the best method.
  - The **Nearest Neighbor (End Insertion)** and **Random Solution** methods were less effective.

Overall, the new methods introduced in this study performed competitively with the best methods from the previous study. The **Greedy 2-Regret Heuristic** showed particularly strong performance, especially in Instance B, where it outperformed the **Greedy Cycle** method.

An anomaly observed is that despite the additional flexibility of adjusting weights in the **Greedy Heuristic with Weighted Sum Criterion**, it did not consistently outperform the **Greedy 2-Regret Heuristic**. This suggests that the choice of weights is critical and may require further tuning or adaptive strategies to achieve optimal performance across different instances.

6 2D Visualization of the Best Solutions

For each method and instance, we generated 2D visualizations of the best solutions. The figures below display the best solution for each method and instance, including both previous and new methods.

6.1 Instance A Best Solutions

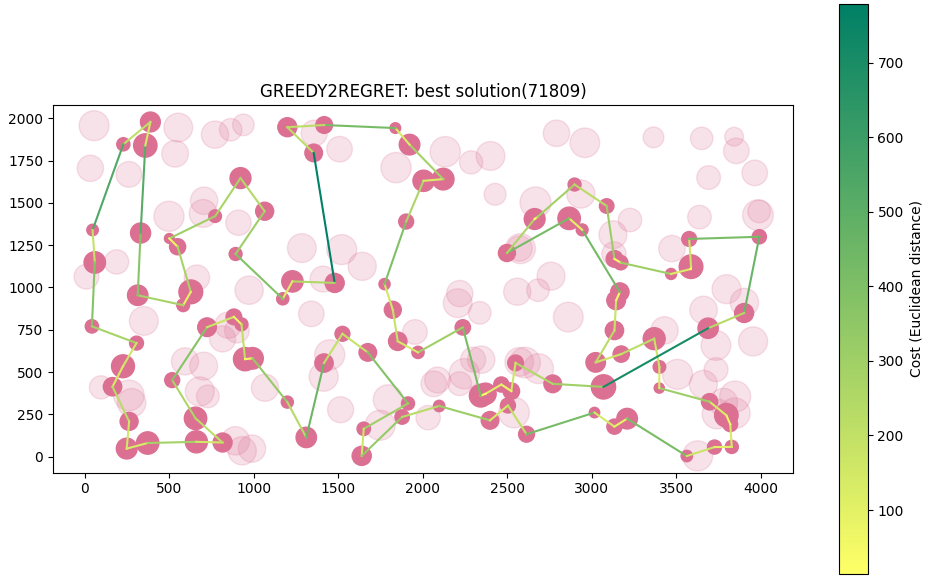

Figure 1: Best solution for Greedy 2-Regret Heuristic on Instance A (Fitness: 71,809).

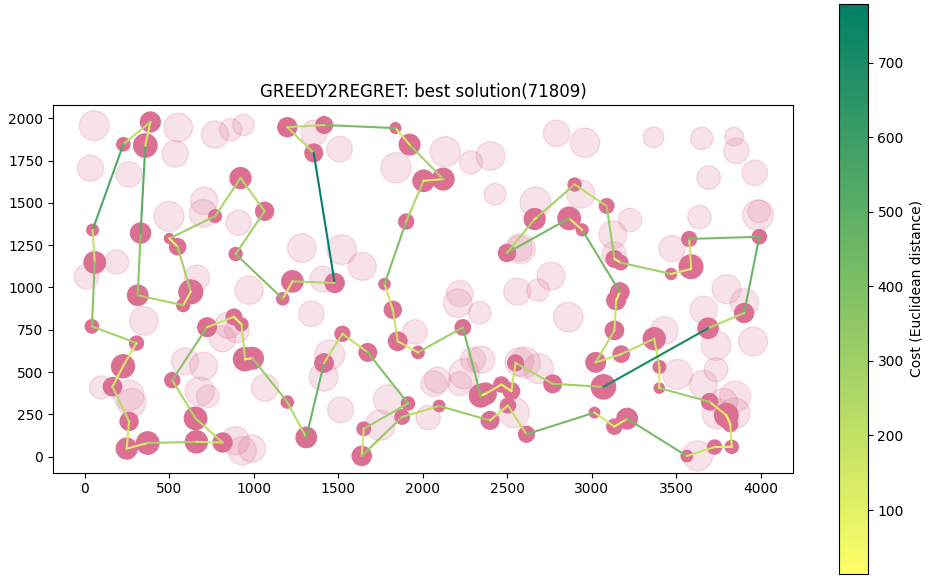

Figure 2: Best solution for Greedy Heuristic with Weighted Sum Criterion on Instance A (Fitness: 71,544).

6.2 Instance B Best Solutions

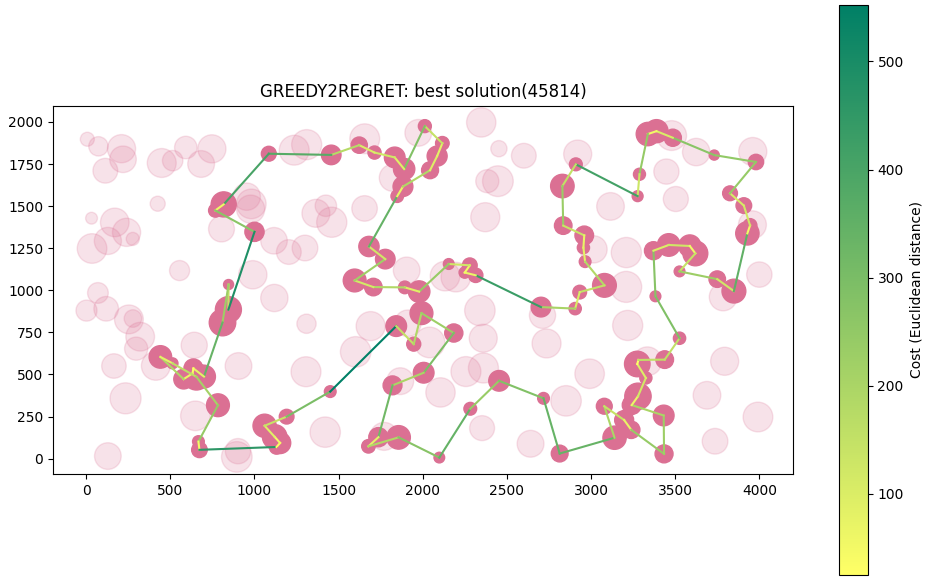

Figure 3: Best solution for Greedy 2-Regret Heuristic on Instance B (Fitness: 45,814).

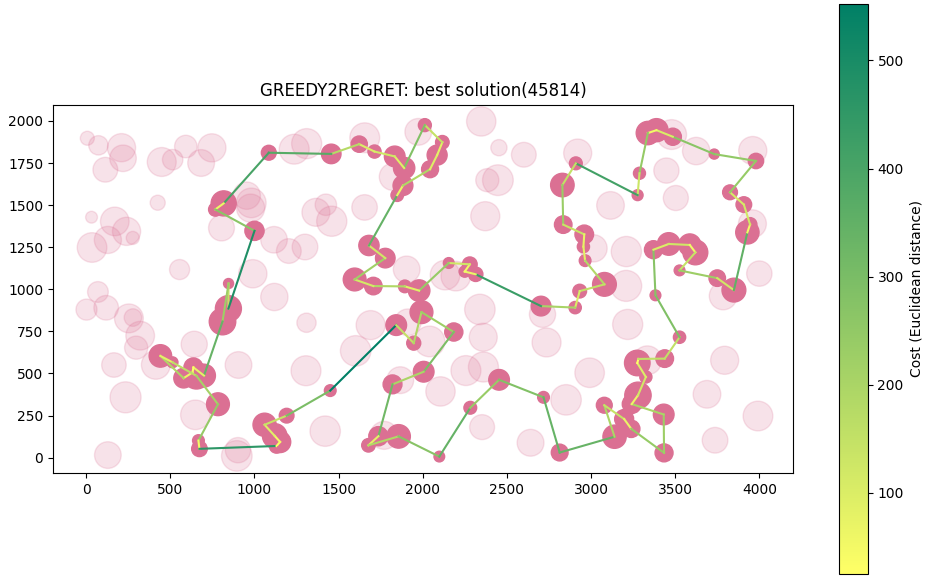

Figure 4: Best solution for Greedy Heuristic with Weighted Sum Criterion on Instance B (Fitness: 46,990).

7 Best Solutions for Each Instance and Method

Below is a list of the best solutions for each method and instance, presented as a list of node indices.

7.1 Instance A Best Solutions

7.1.1 Results for Greedy 2-Regret Heuristic

Best Fitness: 71,809

82, 2, 152, 1, 97, 26, 100, 94, 63, 79, 80, 176, 137, 23, 186, 89, 183, 143, 117, 0, 51, 118, 59, 115, 46, 68, 139, 193, 41, 5, 42, 181, 159, 69, 108, 18, 22, 146, 34, 160, 48, 54, 177, 10, 190, 4, 112, 84, 184, 43, 116, 65, 131, 149, 123, 127, 162, 151, 133, 180, 135, 70, 154, 53, 86, 101, 75, 120, 44, 25, 16, 171, 175, 113, 56, 31, 78, 145, 179, 92, 129, 57, 55, 52, 49, 102, 148, 9, 62, 144, 14, 178, 106, 185, 40, 119, 165, 90, 81, 196

Worst Fitness: 76,323

83, 89, 183, 186, 23, 137, 176, 80, 79, 63, 94, 124, 152, 97, 1, 101, 2, 120, 129, 55, 49, 102, 148, 9, 62, 144, 14, 3, 178, 106, 165, 90, 81, 196, 40, 119, 185, 52, 57, 92, 179, 145, 78, 56, 113, 175, 171, 16, 25, 44, 75, 86, 26, 100, 53, 180, 154, 70, 135, 133, 151, 162, 51, 59, 118, 115, 46, 68, 139, 193, 41, 5, 42, 181, 159, 69, 108, 18, 22, 146, 34, 160, 48, 54, 30, 177, 10, 190, 184, 43, 116, 65, 131, 149, 123, 127, 112, 4, 84, 35

Average Fitness: 73,731.69

7.1.2 Results for Greedy Heuristic with Weighted Sum Criterion

Best Fitness: 71,544

15, 148, 9, 62, 102, 49, 178, 106, 52, 55, 57, 92, 129, 152, 97, 1, 26, 100, 94, 63, 79, 80, 176, 137, 23, 186, 89, 183, 143, 117, 0, 51, 118, 59, 115, 46, 68, 139, 193, 41, 5, 42, 181, 159, 69, 108, 18, 22, 146, 34, 160, 48, 54, 177, 10, 190, 4, 112, 84, 184, 43, 116, 65, 131, 149, 123, 127, 162, 151, 133, 180, 135, 70, 154, 53, 86, 101, 75, 2, 120, 44, 25, 16, 171, 175, 113, 56, 31, 78, 145, 179, 185, 119, 40, 196, 81, 90, 165, 14, 144

Worst Fitness: 76,341

188, 31, 56, 113, 175, 171, 16, 78, 145, 179, 92, 57, 52, 185, 119, 40, 196, 81, 90, 165, 106, 178, 14, 144, 62, 9, 148, 102, 49, 55, 129, 25, 44, 120, 2, 75, 86, 101, 1, 97, 152, 26, 100, 53, 154, 70, 135, 180, 133, 79, 63, 94, 80, 176, 137, 23, 186, 89, 183, 143, 117, 0, 51, 151, 162, 127, 123, 131, 149, 59, 118, 65, 116, 43, 184, 84, 112, 4, 190, 10, 177, 54, 48, 160, 34, 181, 42, 5, 115, 46, 68, 139, 41, 193, 159, 146, 22, 18, 69, 108

Average Fitness: 73,762.84

7.2 Instance B Best Solutions

7.2.1 Results for Greedy 2-Regret Heuristic

Best Fitness: 45,814

112, 121, 19, 73, 54, 31, 193, 198, 117, 164, 136, 190, 80, 175, 78, 142, 45, 5, 177, 104, 8, 144, 111, 82, 21, 61, 36, 91, 141, 77, 81, 153, 187, 165, 163, 89, 127, 137, 114, 103, 26, 113, 180, 176, 194, 166, 86, 95, 130, 99, 22, 185, 179, 172, 66, 94, 47, 148, 60, 20, 28, 149, 4, 140, 183, 152, 170, 34, 55, 18, 62, 128, 124, 106, 143, 35, 0, 109, 29, 160, 33, 138, 182, 11, 139, 168, 195, 145, 15, 3, 70, 13, 132, 169, 188, 6, 147, 191, 90, 51

Worst Fitness: 59,121

178, 10, 133, 115, 147, 71, 120, 51, 98, 74, 118, 116, 121, 112, 19, 151, 73, 164, 54, 31, 136, 45, 5, 177, 21, 87, 104, 182, 138, 139, 168, 195, 145, 15, 189, 152, 184, 155, 3, 70, 161, 13, 132, 169, 188, 6, 134, 43, 11, 49, 33, 160, 29, 109, 35, 0, 111, 144, 56, 8, 82, 91, 77, 81, 153, 163, 103, 89, 127, 187, 146, 97, 141, 79, 61, 36, 142, 78, 175, 162, 80, 190, 193, 117, 198, 156, 1, 16, 27, 38, 131, 125, 191, 90, 122, 135, 32, 96, 63, 100, 72, 17, 107, 40

Average Fitness: 50,794.27

7.2.2 Results for Greedy Heuristic with Weighted Sum Criterion

Best Fitness: 46,990

164, 73, 136, 80, 190, 193, 31, 54, 117, 198, 156, 1, 16, 27, 38, 135, 63, 122, 131, 121, 51, 125, 90, 191, 147, 6, 188, 169, 132, 13, 70, 3, 15, 145, 195, 168, 139, 11, 182, 138, 33, 160, 29, 0, 109, 35, 111, 144, 56, 104, 8, 82, 21, 177, 5, 45, 142, 78, 175, 61, 36, 91, 141, 77, 81, 153, 187, 163, 89, 127, 137, 114, 103, 113, 180, 176, 194, 166, 86, 185, 179, 94, 47, 148, 60, 20, 22, 99, 130, 95, 183, 140, 152, 34, 55, 18, 62, 124, 106, 143

Worst Fitness: 58,454

67, 71, 191, 90, 125, 131, 38, 27, 16, 1, 156, 198, 117, 193, 80, 190, 162, 175, 78, 142, 5, 177, 21, 87, 104, 182, 138, 139, 168, 195, 145, 15, 3, 70, 161, 13, 132, 169, 188, 6, 134, 43, 11, 49, 33, 160, 29, 109, 35, 0, 111, 144, 56, 8, 82, 91, 77, 81, 153, 163, 103, 89, 127, 187, 146, 97, 141, 79, 61, 36, 45, 136, 31, 54, 164, 73, 151, 19, 112, 121, 118, 74, 98, 51, 120, 147, 115, 178, 10, 133, 122, 135, 32, 102, 63, 100, 72, 17, 107, 40

Average Fitness: 50,992.64

8 Solution Verification with the Solution Checker

We verified the correctness of the best solutions for each method and instance using an Excel-based solution checker. The fitness values calculated by our algorithms matched exactly with those obtained from the solution checker, confirming the validity of our implementations.

Table 3: Fitness Verification for All Methods

Instance	Method	Calculated Fitness	Checker Fitness
A	Greedy 2-Regret Heuristic	71,809	71,809
A	Greedy Weighted Sum Heuristic	71,544	71,544
B	Greedy 2-Regret Heuristic	45,814	45,814
B	Greedy Weighted Sum Heuristic	46,990	46,990

9 Source Code

The complete source code for this project, including all implemented algorithms and scripts, is available on GitHub:

https://github.com/notbulubula/EvolutionaryComputation

10 Conclusion

The extension of the study to include the **Greedy 2-Regret Heuristic** and the **Greedy Heuristic with Weighted Sum Criterion** has shown that these methods can produce competitive results compared to the previously implemented heuristics. The ability to incorporate regret and adjust weights allows for more adaptable algorithms that can be tailored to specific problem instances.

The **Nearest Neighbor (Flexible Insertion)** method consistently achieved the best fitness values across both instances, suggesting that it is particularly effective for the types of datasets used in this study. The **Greedy 2-Regret Heuristic** also demonstrated strong performance, especially in Instance B.

Future research could explore optimizing the weights used in the weighted sum heuristic, potentially using metaheuristic algorithms to find the best weight configurations for different datasets. Additionally, combining the strengths of multiple heuristics through hybrid methods may further enhance performance.