POZNAN UNIVERSITY OF TECHNOLOGY

FACULTY OF COMPUTING AND TELECOMMUNICATION
Institute of Computing Science

LABORATORY 9: HYBRID EVOLUTIONARY ALGORITHM

Piotr Balewski, 156037
Lidia Wiśniewska, 156063

POZNAŃ 2025

1 Problem Description

The problem description remains the same as in previous labs. The problem is to select exactly 50% of the nodes (from n available) and build a Hamiltonian cycle through the selected nodes. The objective is to minimize the sum of two components:

1. The total length of the Hamiltonian cycle (calculated as the sum of Euclidean distances between consecutive nodes in the cycle).
2. The total cost of the selected nodes (the sum of the cost attributes for each selected node).

All distances are rounded to the nearest integer.

2 Hybrid Evolutionary Algorithm (HEA)

The goal of this laboratory was to implement a Hybrid Evolutionary Algorithm, which combines the global exploration capabilities of evolutionary computation with the exploitation power of local search. As instructed, we implemented the steady-state version with an elite population set to 20, without copies of the solutions in the population and random (uniform) selection of the parents. The time limit for the algorithm was set equal to the average execution time of the MSLS algorithm from lab 6.

```
procedure run_HEA(instance, time_limit, useLocalSearch, operatorType)
    population = getInitialPopulation(size=20) //random solution + LS

    while time < time_limit do
        // A. Selection (Uniform)
        p1 = selectRandom(population)
        p2 = selectRandom(population)

        // B. Recombination
        if operatorType == 1
            offspring = operator1(p1, p2)
        else
            offspring = operator2(p1, p2)

        // C. Local Search
        if useLocalSearch
            offspring = localSearch(offspring)

        // D. Steady State Update
        worst = getWorst(population)
        if cost(offspring) < cost(worst) AND !isDuplicate(offspring, population)
            replace(worst, offspring)

    return getBest(population)
```

2.1 Recombination Operators

Two distinct recombination operators were implemented to combine traits from parent solutions.

Operator 1

This operator focuses on preserving common subpaths.

```
procedure operator1(p1, p2, length=100)
    //1. get common segments
    segments = getCommonEdges(p1, p2) // e.g. [[i,j,k], [o,p],...]
    used_nodes = nodes present in segments

    common_nodes = getCommonNodes(p1, p2)
    for node in common_nodes
        if node not in used_nodes
            //add to segments and used_nodes

    //2. fill missing #nodes
    available_nodes = getRemaining(used_nodes)
    shuffle(availabe_nodes)
    needed = length - used_nodes.size()
    while(needed != 0)
        //add to the segments random node from available_nodes

    //3. Combine randomly
    shuffle(segments)
    child = ArrayList

    for seg in segments
        if seg.size > 1 and randBoolean()
            reverse(seg)
        child.addAll(seg)

    return child
```

Operator 2

This operator is based on the logic of filter and repair.

```
procedure operator2(p1, p2)
    child = getCommonNodes(p1,p2)
    return repair(child, regretWeight=0.5)
```

We tested this operator in two variants: with and without a subsequent Local Search step.

3 Experimental Results

Each configuration was run 20 times per instance with a time limit equivalent to the average MSLS runtime.

3.1 TSPA: HEA1 + LS

Results: min: 69464 max: 70553 avg: 70054
Best Path: [59, 118, 51, 151, 162, 149, 131, 65, 116, 43, 42, 181, 34, 160, 48, 54, 177, 10, 190, 184, 84, 4, 112, 123, 127, 70, 135, 154, 180, 53, 26, 100, 86, 75, 101, 1, 97, 152, 2, 120, 44, 25, 16, 171, 175, 113, 56, 31, 157, 196, 81, 90, 27, 165, 119, 40, 185, 179, 145, 78, 78, 92, 129, 57, 55, 52, 106, 178, 49, 14, 144, 102, 9, 62, 148, 124, 94, 63, 79, 133, 80, 176, 137, 23, 89, 183, 143, 0, 117, 93, 140, 108, 18, 22, 159, 193, 41, 139, 68, 46, 115]

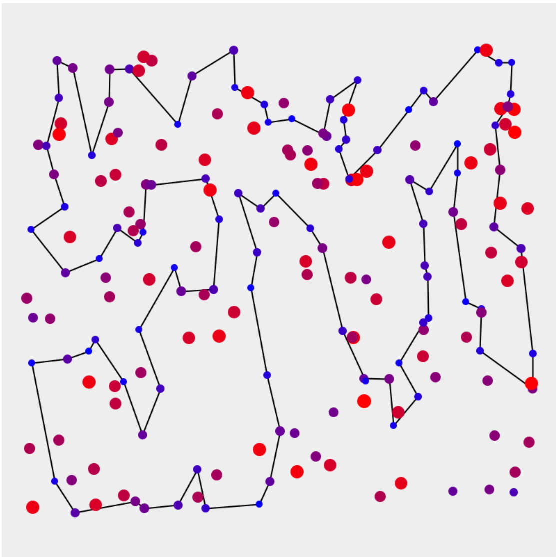

Figure 1: Best solution found by HEA1 + LS for TSPA.

3.2 TSPA: HEA2 + LS

Results: min: 70124 max: 71436 avg: 70759
Best Path: [9, 148, 124, 94, 63, 79, 80, 133, 151, 162, 59, 118, 51, 176, 137, 23, 89, 183, 143, 0, 117, 93, 140, 68, 46, 139, 115, 193, 41, 159, 69, 108, 18, 22, 146, 34, 48, 54, 177, 10, 190, 4, 112, 84, 35, 184, 160, 181, 42, 43, 116, 65, 131, 149, 123, 127, 135, 70, 180, 154, 53, 26, 100, 97, 152, 1, 101, 86, 75, 2, 120, 44, 25, 129, 57, 92, 179, 145, 78, 16, 171, 175, 113, 56, 31, 157, 196, 81, 90, 165, 40, 185, 52, 55, 178, 106, 49, 14, 144, 62]

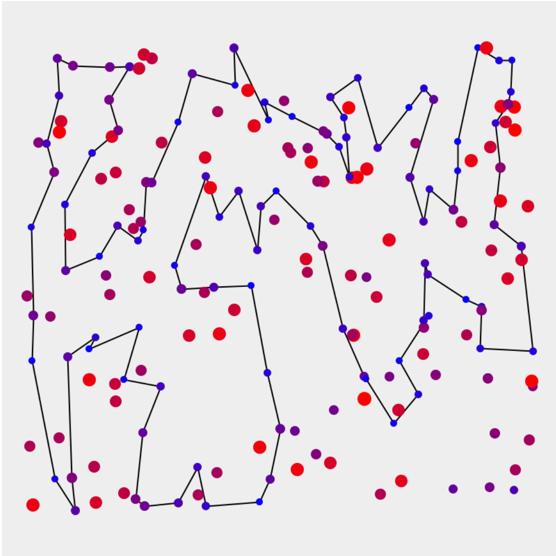

Figure 2: Best solution found by HEA2 + LS for TSPA.

3.3 TSPA: HEA2

Results: min: 70481 max: 71458 avg: 71097

Best Path: [14, 144, 9, 62, 148, 124, 94, 63, 79, 80, 176, 137, 23, 89, 183, 143, 0, 117, 93, 140, 108, 18, 22, 146, 34, 160, 48, 54, 10, 177, 184, 84, 4, 112, 149, 131, 116, 65, 43, 42, 181, 159, 41, 193, 139, 68, 46, 115, 59, 118, 51, 151, 133, 162, 123, 127, 70, 135, 154, 180, 53, 121, 26, 100, 97, 152, 101, 1, 86, 75, 2, 129, 92, 57, 55, 52, 179, 145, 78, 25, 44, 120, 16, 171, 175, 113, 56, 31, 157, 196, 81, 90, 27, 39, 165, 185, 40, 106, 178, 49]

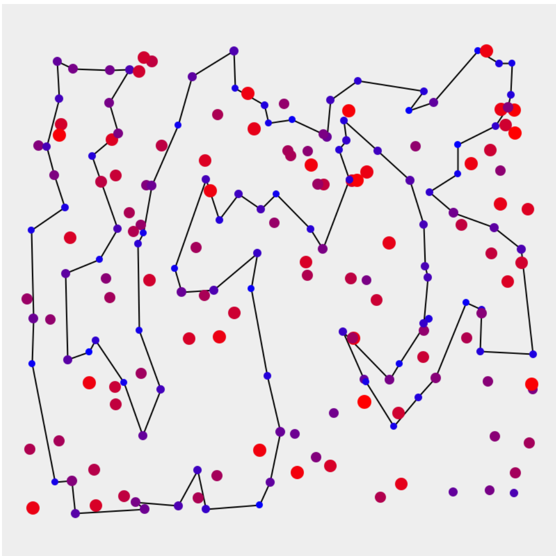

Figure 3: Best solution found by HEA2 for TSPA.

3.4 TSPB: HEA1 + LS

Results: min: 43763 max: 44528 avg: 44097
Best Path: [62, 18, 55, 34, 170, 152, 183, 140, 4, 149, 28, 20, 60, 148, 47, 94, 66, 179, 185, 99, 130, 95, 86, 166, 194, 176, 113, 103, 127, 89, 163, 153, 81, 77, 141, 91, 61, 36, 177, 5, 78, 175, 142, 45, 80, 190, 136, 73, 54, 31, 193, 117, 198, 1, 16, 27, 38, 63, 40, 107, 10, 133, 122, 135, 131, 121, 51, 90, 191, 147, 6, 188, 169, 132, 70, 3, 15, 145, 13, 195, 168, 43, 139, 11, 138, 33, 160, 144, 104, 8, 21, 82, 111, 29, 0, 109, 35, 143, 106, 124]

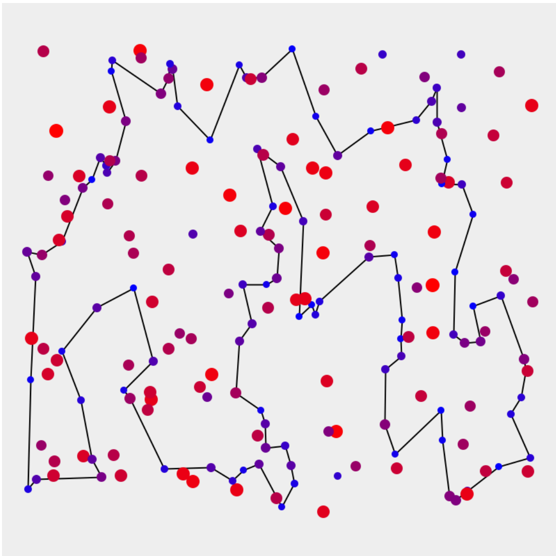

FIGURE 4: Best solution found by HEA1 + LS for TSPB.

3.5 TSPB: HEA2 + LS

Results: min: 44192 max: 45680 avg: 44857
Best Path: [0, 109, 35, 143, 106, 124, 62, 55, 18, 34, 170, 152, 183, 140, 4, 149, 28, 20, 60, 148, 47, 94, 66, 179, 185, 99, 130, 95, 86, 166, 194, 176, 113, 103, 114, 137, 127, 89, 163, 187, 153, 81, 77, 141, 91, 61, 36, 177, 5, 78, 175, 45, 190, 80, 136, 73, 54, 31, 193, 117, 198, 156, 1, 27, 38, 63, 135, 131, 121, 51, 90, 122, 107, 40, 133, 10, 147, 6, 188, 169, 13, 132, 70, 3, 15, 145, 168, 195, 43, 139, 11, 138, 33, 160, 144, 104, 8, 82, 111, 29]

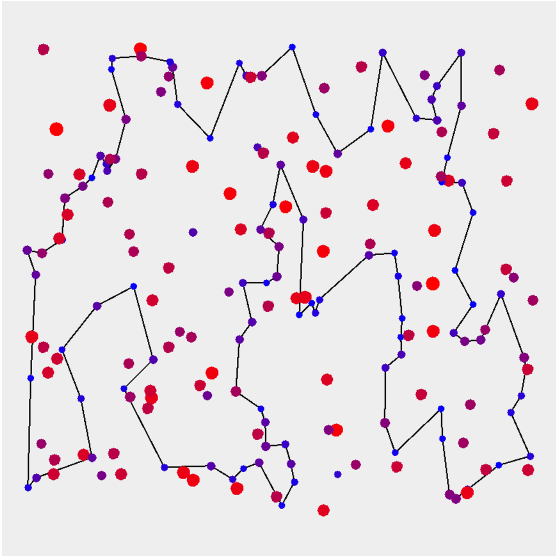

FIGURE 5: Best solution found by HEA2 + LS for TSPB.

3.6 TSPB: HEA2

Results: min: 44463 max: 46364 avg: 45041
Best Path: [29, 0, 109, 35, 143, 106, 124, 62, 18, 55, 34, 170, 152, 183, 140, 4, 149, 28, 20, 60, 47, 148, 94, 66, 57, 172, 179, 185, 99, 130, 95, 86, 166, 194, 176, 113, 103, 163, 89, 127, 165, 187, 153, 81, 77, 141, 91, 61, 36, 177, 5, 175, 78, 162, 190, 80, 136, 73, 31, 193, 54, 117, 198, 156, 1, 27, 38, 135, 63, 40, 107, 133, 122, 131, 121, 90, 51, 147, 6, 188, 169, 132, 13, 3, 70, 15, 145, 195, 168, 43, 139, 11, 138, 33, 160, 144, 104, 8, 82, 111]

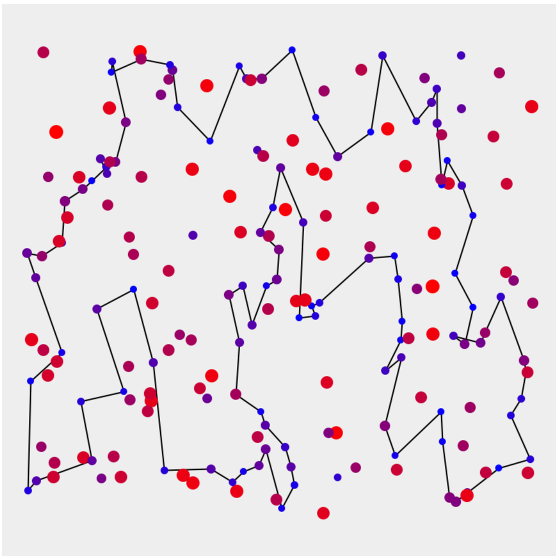

FIGURE 6: Best solution found by HEA2 for TSPB.

Table 1: Results Summary

TSPA

Method	Avg	Min	Max
MSLS	72505	71431	73209
ILS	69462	69176	70196
LNS	69682	69273	70154
LNS+LS	69454	69207	69865
HEA1 + LS	70054	69464	70553
HEA2 + LS	70759	70124	71436
HEA2	71097	70481	71458

TSPB

Method	Avg	Min	Max
MSLS	46766	46040	47280
ILS	43819	43535	44395
LNS	44193	43630	45268
LNS+LS	43972	43636	44749
HEA1 + LS	44097	43763	44528
HEA2 + LS	44857	44192	45680
HEA2	45041	44463	46364

4 Conclusions

The HEA1, which preserves common edges and nodes outperformed the HEA2, which focused only on common nodes. Additionally, the HEA2 variant without Local Search performs noticeably worse, confirming that Local Search handles well the necessary refinement after recombination.

Solution Checker: All best solutions obtained were verified using the provided solution checker.
Source Code: https://github.com/PBalewski/EvolutionaryComputation/tree/main/lab9