POZNAN UNIVERSITY OF TECHNOLOGY

FACULTY OF COMPUTING AND TELECOMMUNICATION
Institute of Computing Science

LABORATORY 6: MULTIPLE START LOCAL SEARCH (MSLS) AND ITERATED LOCAL SEARCH (ILS)

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 Extended Local Search Methods

The goal of this assignment is to overcome the main limitation of standard Local Search (LS), which is getting stuck in local optima. In order to test that, we implement two approaches: Multiple Start Local Search (MSLS) and Iterated Local Search (ILS). Both methods utilize the basic Steepest Local Search with edge exchange developed in previous assignments as a subroutine.

2.1 Multiple Start Local Search (MSLS)

In MSLS instead of running the local search once from a single random starting point, the algorithm restarts the search procedure multiple times from different random initial solutions. This increases the coverage of the search space and improves the probability of finding a better local optimum (or the global optimum).

In this experiment, MSLS performs 200 iterations of the LS. In each iteration, a random solution is generated, and the steepest local search is run until convergence. The final result of a single run of MSLS is the best solution found across all 200 iterations.

```
procedure runMSLS(instance, iterations)
    best_global = NULL

    for i=0 to iterations do
        initial_solution = getRandomSolution(instance)
        result = localSearch(initial_solution, instance)

        if best_global is null or result.cost < best_global.cost
            best_global = result

    return best_global
```

2.2 Iterated Local Search (ILS)

Iterated Local Search attempts to improve upon a local optimum not by restarting randomly, but by perturbing the current best solution. The logic is that a good local optimum may be close to other, potentially better, local optima. The stopping condition for ILS in this experiment is the average duration for one MSLS run.

```
procedure Run_ILS(instance, time_limit)
    start_time = getCurrentTime()

    initial_solution = getRandomSolution(instance)
    best_solution = localSearch(initial_solution, instance)

    ls_runs_count = 1

    while (getCurrentTime() - start_time) < time_limit do
        // 1. Perturbation
        perturbed_solution = perturb(best_solution)

        // 2. Local Search
        result = localSearch(perturbed_solution, instance)
        ls_runs_count++

        // 3. Acceptance (Better or Equal)
        if result.cost < best_solution.cost
            //update

    return best_solution, ls_runs_count
```

2.3 Perturbation Strategy

The perturbation strategy employed in the Iterated Local Search (ILS) is a segment swap. This method divides the current solution into four disjoint segments, denoted as A, B, C, and D. The perturbation reconstructs the solution by swapping the order of the two middle segments, changing the sequence from $A \to B \to C \to D$ to $A \to C \to B \to D$. The algorithm stochastically selects three cut points p_1, p_2, p_3, ensuring that every segment has a length of at least one node.

```
procedure perturb(solution, n)
    // 1. select cut points ensuring space for remaining segments

    // p1: random range [1, n-3]
    p1 = 1 + random[0, n-3)

    // p2: random range [p1+1, n-2]
    p2 = p1+1 + random[0, n-2-p1)

    // p3: random range [p2+1, n-1]
    p3 = p2+1 + random[0, n-1-p2)

    // 2. get segments
    A = solution[0, p1)
    B = solution[p1, p2)
    C = solution[p2, p3)
    D = solution[p3, n)

    // 3. reconstruct swapped
    new_solution = concatenate(A, C, B, D)

return new_solution
```

3 Experimental Results

The computational experiment involved running both MSLS and ILS 20 times for each instance (TSPA and TSPB).

3.1 TSPA: MSLS

Results: min: 71431 max: 72505 avg: 73209
Best Path: [139, 41, 193, 159, 22, 18, 108, 93, 117, 0, 170, 143, 183, 89, 137, 176, 80, 51, 118, 59, 116, 65, 47, 43, 131, 149, 162, 151, 133, 79, 63, 94, 124, 167, 148, 9, 62, 49, 144, 14, 138, 106, 178, 55, 52, 185, 119, 40, 165, 90, 81, 196, 179, 57, 129, 92, 145, 78, 31, 113, 175, 171, 16, 44, 25, 120, 2, 75, 86, 101, 1, 152, 97, 26, 100, 53, 154, 180, 135, 70, 127, 123, 112, 35, 4, 84, 190, 10, 177, 54, 48, 184, 160, 34, 181, 42, 5, 115, 46, 68]

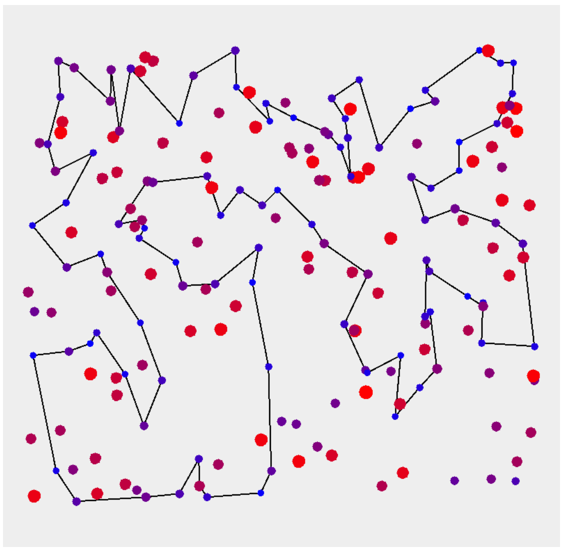

FIGURE 1: Best solution found by MSLS for TSPA.

3.2 TSPA: ILS

Results: min: 69176 max: 69462 avg: 70196

Best Path: [9, 148, 124, 94, 63, 79, 80, 176, 137, 23, 186, 89, 183, 143, 0, 117, 93, 140, 108, 18, 22, 159, 193, 41, 139, 68, 46, 115, 59, 118, 51, 151, 133, 162, 149, 131, 65, 116, 43, 42, 181, 34, 160, 48, 54, 177, 10, 190, 184, 35, 84, 4, 112, 123, 127, 70, 135, 154, 180, 53, 100, 26, 86, 75, 101, 1, 97, 152, 2, 120, 44, 25, 16, 171, 175, 113, 56, 31, 78, 145, 196, 81, 90, 165, 119, 40, 185, 179, 92, 129, 57, 55, 52, 106, 178, 49, 14, 144, 102, 62]

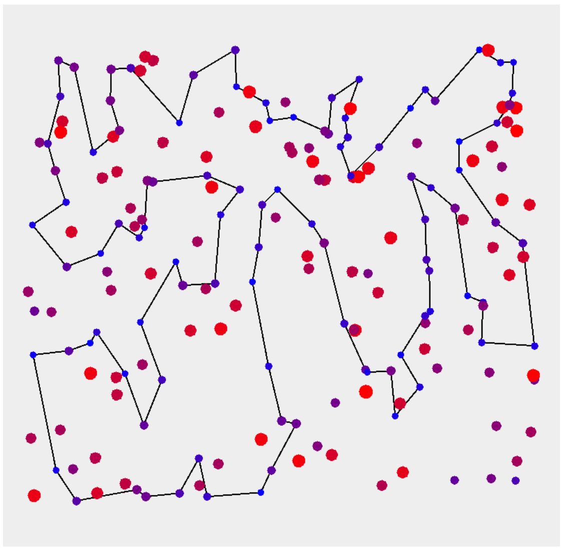

Figure 2: Best solution found by ILS for TSPA.

3.3 TSPB: MSLS

Results: min: 46040 max: 46766 avg: 47280

Best Path: [188, 169, 132, 13, 168, 195, 145, 15, 3, 70, 155, 152, 170, 34, 55, 18, 62, 106, 124, 128, 95, 99, 130, 183, 140, 28, 20, 60, 47, 148, 94, 179, 185, 86, 166, 194, 176, 180, 113, 103, 127, 89, 163, 187, 153, 77, 97, 141, 36, 61, 21, 8, 104, 144, 111, 35, 0, 109, 12, 29, 160, 33, 138, 11, 139, 182, 25, 177, 5, 78, 175, 80, 190, 136, 73, 164, 54, 31, 193, 198, 117, 1, 27, 38, 131, 121, 51, 191, 90, 122, 135, 102, 63, 40, 107, 133, 10, 178, 147, 6]

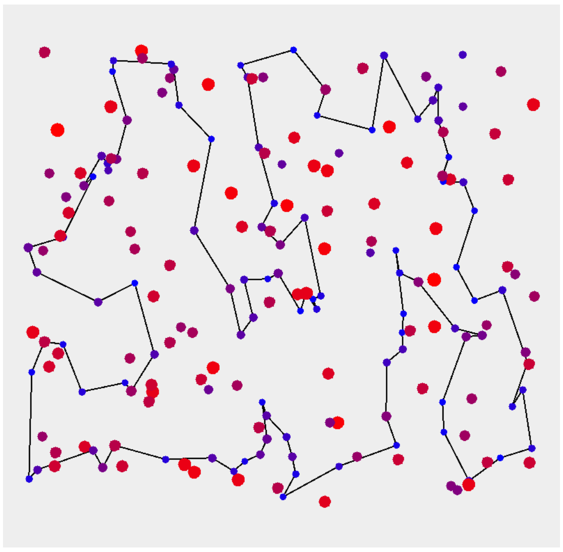

Figure 3: Best solution found by MSLS for TSPB.

3.4 TSPB: ILS

Results: min: 43535 max: 43819 avg: 44395
Best Path: [21, 82, 111, 144, 33, 160, 29, 0, 109, 35, 143, 106, 124, 62, 18, 55, 34, 170, 152, 183, 140, 4, 149, 28, 20, 60, 148, 47, 94, 66, 179, 22, 99, 130, 95, 185, 86, 166, 194, 176, 113, 114, 137, 127, 89, 103, 163, 187, 153, 81, 77, 141, 91, 61, 36, 177, 5, 78, 175, 142, 45, 80, 190, 136, 73, 54, 31, 193, 117, 198, 156, 1, 38, 63, 40, 107, 133, 122, 135, 131, 121, 51, 90, 147, 6, 188, 169, 132, 70, 3, 15, 145, 13, 195, 168, 139, 11, 138, 104, 8]

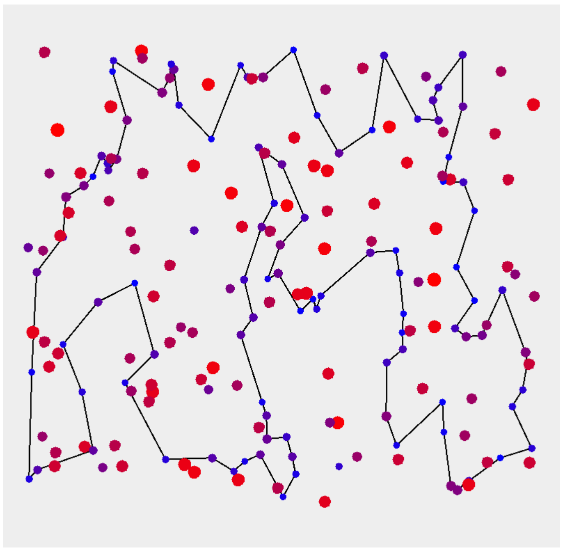

Figure 4: Best solution found by ILS for TSPB.

Table 1: Results summary

TSPA

Method	Avg	Min	Max
Steepest Edge	75135	72495	79990
MSLS	72505	71431	73209
ILS	69462	69176	70196
ILS (LS Runs)	5226	5037	5414

TSPB

Method	Avg	Min	Max
Steepest Edge	49275	46671	52022
MSLS	46766	46040	47280
ILS	43819	43535	44395
ILS (LS Runs)	5471	5211	5667

Table 2: Execution times (in milliseconds) summary

TSPA

Method	Avg	Min	Max
Steepest Edge	48,43	40,19	173,47
MSLS	8903,23	8819,11	9225,86
ILS	8904,26	8903,27	8906,19

TSPB

Method	Avg	Min	Max
Steepest Edge	47,53	38,74	179,38
MSLS	8992,33	8872,70	9184,66
ILS	8993,30	8992,44	8994,44

4 Conclusions

The experimental results demonstrate that extending the standard Steepest Local Search with restart or perturbation mechanisms improves the quality of the found solutions. However, direct comparison reveals that ILS is the superior metaheuristic for this problem, consistently achieving the lowest objective function for both TSPA and TSPB instances at a similar time.

Solution Checker: All best solutions obtained were verified using the provided solution checker.
Source Code: https://github.com/PBalewski/EvolutionaryComputation/tree/main/lab6