

# Assignment 6 - Multiple Start and Iterated Local Search for Selective TSP

---

## Authors

---

- Mateusz Idziejczak 155842
- Mateusz Stawicki 155900

## Github

---

<https://github.com/Luncenok/EvolutionaryComputing>

## Problem Description

---

This is the same variant of the Traveling Salesman Problem as in previous assignments:

- Select exactly 50% of nodes (rounded up if odd)
- Form a Hamiltonian cycle through selected nodes
- Minimize: total path length + sum of selected node costs
- Distances are Euclidean distances rounded to integers

Instances:

- **TSPA, TSPB** with 200 nodes, selecting 100 nodes.

## Goal

---

Implement and compare two iterative local search methods:

1. **Multiple Start Local Search (MSLS)**: Run local search from multiple random starting solutions and keep the best result.
2. **Iterated Local Search (ILS)**: Start from a random solution, apply local search, then iteratively perturb the solution and apply local search again.

For both methods:

- Use the best performing local search from Assignment 3: **Random + Steepest + Edges**
- Run **20 times** per instance to collect statistics
- For MSLS: perform **200 iterations** (local search runs) per execution
- For ILS: use a **time limit equal to the average MSLS time**, report the number of local search runs

## Algorithm Pseudocode

---

### Local Search Base

Both MSLS and ILS use the same local search as their core optimization component: **Steepest Descent with Edge Exchange** from Assignment 3.

This local search performs:

- **Neighborhood:** 2-opt edge exchanges
- **Strategy:** Steepest descent (select best improving move in each iteration)
- Terminates when no improving move exists (local optimum)

## Multiple Start Local Search (MSLS)

MSLS repeatedly applies local search from different random starting solutions and keeps the best result found.

```
MSLS(iterations):  
    bestSolution = null  
    bestObjective = ∞  
  
    for i = 1 to iterations:  
        # Generate random starting solution  
        initial = generateRandomSolution()  
  
        # Apply local search to find local optimum  
        localOptimum = localSearch(initial)  
  
        # Evaluate solution quality  
        obj = objective(localOptimum)  
  
        # Keep best solution found  
        if obj < bestObjective:  
            bestObjective = obj  
            bestSolution = localOptimum  
  
    return bestSolution
```

## Iterated Local Search (ILS)

ILS starts from one solution, applies local search, then iteratively perturbs the solution and searches again within a time limit.

```

ILS(timeLimit):
    # Initialize with random solution
    current = generateRandomSolution()
    current = localSearch(current)

    bestSolution = current
    bestObjective = objective(current)

    startTime = now()

    while (now() - startTime) < timeLimit:
        # Perturb current solution to escape local optimum
        perturbed = perturb(current)

        # Apply local search to perturbed solution
        candidate = localSearch(perturbed)
        obj = objective(candidate)

        # Update best solution if improved
        if obj < bestObjective:
            bestObjective = obj
            bestSolution = candidate

        # Always continue from new solution (exploration)
        current = candidate

    return bestSolution

```

## Perturbation Strategy

The perturbation function is crucial for ILS effectiveness. We use a combination of moves:

```

perturb(solution):
    perturbed = copy(solution)
    k = adaptivePerturbationStrength(solution) # typically 2-5

    # Apply multiple random 2-opt moves to change tour structure
    for i = 1 to k:
        pos1, pos2 = selectTwoNonAdjacentPositions()
        perturbed = apply2Opt(perturbed, pos1, pos2)

    # Occasionally exchange a selected node with an unselected node
    if random() < 0.3:
        nodeInTour = selectRandomPosition(perturbed)
        nodeNotInTour = selectRandomUnselectedNode()
        perturbed[nodeInTour] = nodeNotInTour

    return perturbed

```

**Perturbation rationale:**

- Multiple 2-opt moves escape local optima by changing tour structure
- Node exchanges allow exploration of different node selections (crucial for Selective TSP)
- Perturbation strength (k) scales with problem size
- Combination balances exploration (escaping basin) and exploitation (staying in promising regions)

## Experimental Setup

- **Instances:** TSPA, TSPB (200 nodes, 100 selected)
- **Objective:** Minimize path length + sum of selected node costs
- **Local search:** Random + Steepest + Edges (from Assignment 3)
- **Evaluation:**
  - Run both MSLS and ILS **20 times** per instance
  - For MSLS: each run performs **200 local search iterations**
  - For ILS: each run uses a **time limit = average MSLS time**
  - Report min, max, and average objective values and running times
  - For ILS: also report average number of local search runs

**Experimental protocol:**

1. Run MSLS 20 times (200 iterations each) and collect statistics
2. Calculate average MSLS time
3. Run ILS 20 times with time limit = average MSLS time
4. Compare solution quality and efficiency

## Key Results

**TSPA Instance**

Method	Min Obj	Max Obj	Avg Obj	Min Time (ms)	Max Time (ms)	Avg Time (ms)	Avg LS Runs
MSLS (200 iter)	70937	70937	70937	3223.82	3482.74	3304.00	200
ILS (time limit)	69141	69476	69305	3304.07	3305.80	3304.60	3474.6

**TSPB Instance**

Method	Min Obj	Max Obj	Avg Obj	Min Time (ms)	Max Time (ms)	Avg Time (ms)	Avg LS Runs
MSLS (200 iter)	45799	45799	45799	3210.38	3381.45	3246.32	200
ILS (time limit)	43456	44291	43613	3246.40	3247.58	3246.79	3543.55

## Summary Comparison

Instance	MSLS Avg	ILS Avg	Improvement	MSLS Time	ILS Time	LS Runs
TSPA	70937	69305	<b>-2.3%</b>	3304 ms	3304.6 ms	3474.6
TSPB	45799	43613	<b>-4.8%</b>	3246 ms	3246.79 ms	3543.55

### Key observations:

- ILS consistently finds better solutions than MSLS on both instances
- TSPA: 2.3% improvement (1632 units better)
- TSPB: 4.8% improvement (2186 units better)
- ILS completes  $\sim 17\times$  more local search runs in the same time (3474 vs 200 for TSPA)
- Both methods have consistent runtimes with very low variance
- MSLS finds the same solution in all 20 runs (min = max = avg), suggesting early convergence

## Comparison with All Previous Methods

Complete results for all methods tested throughout the course:

### TSPA Instance - Objective Function Values:

Method	Min	Max	Avg
Random	235453	288189	264501
Nearest Neighbor (end only)	83182	89433	85108
Nearest Neighbor (any position)	71179	75450	73178
Greedy Cycle	71488	74410	72646
Greedy 2-Regret	105852	123428	115474
Greedy Weighted (2-Regret + BestDelta)	71108	73395	72129
Nearest Neighbor Any 2-Regret	106373	126570	116659
Nearest Neighbor Any Weighted (2-Regret + BestDelta)	70010	75452	72401
LS Random + Steepest + Nodes	80903	97156	88323
LS Random + Greedy + Nodes	86293	102205	92779
LS Random + Greedy + Edges	75576	86423	81269
LS Greedy + Steepest + Nodes	70626	72950	71614
LS Greedy + Steepest + Edges	70510	72614	71460
LS Greedy + Greedy + Nodes	71093	73048	71913
LS Greedy + Greedy + Edges	70977	72844	71817
LS Random + Steepest + Edges	70937	78033	73945

Method	Min	Max	Avg
LM Random + Steepest + Edges	71993	80945	74973
Candidates + Random + Steepest + Edges (k=5)	78119	91398	84660
Candidates + Random + Steepest + Edges (k=10)	73550	83200	77494
Candidates + Random + Steepest + Edges (k=15)	71917	80679	75268
Candidates + Random + Steepest + Edges (k=20)	71417	79637	74451
LM Candidates + Random + Steepest + Edges (k=10)	72274	79625	74829
LM Candidates + Random + Steepest + Edges (k=20)	71993	80945	74962
<b>MSLS (200 iterations)</b>	<b>70937</b>	<b>70937</b>	<b>70937</b>
<b>ILS (time limit = 3304 ms)</b>	<b>69141</b>	<b>69476</b>	<b>69305</b>

**TSPB Instance - Objective Function Values:**

Method	Min	Max	Avg
Random	189071	238254	212513
Nearest Neighbor (end only)	52319	59030	54390
Nearest Neighbor (any position)	44417	53438	45870
Greedy Cycle	49001	57324	51400
Greedy 2-Regret	66505	77072	72454
Greedy Weighted (2-Regret + BestDelta)	47144	55700	50950
Nearest Neighbor Any 2-Regret	67121	79013	73646
Nearest Neighbor Any Weighted (2-Regret + BestDelta)	44891	55247	47653
LS Random + Steepest + Nodes	56207	70573	63219
LS Random + Greedy + Nodes	58261	72047	65529
LS Random + Greedy + Edges	50177	59362	54184
LS Greedy + Steepest + Nodes	43826	50876	45414
LS Greedy + Steepest + Edges	43921	50629	44979
LS Greedy + Greedy + Nodes	43917	51144	45561
LS Greedy + Greedy + Edges	43845	51072	45371
LS Random + Steepest + Edges	45799	51543	48313
LM Random + Steepest + Edges	46324	53526	49391
Candidates + Random + Steepest + Edges (k=5)	46328	53421	49996
Candidates + Random + Steepest + Edges (k=10)	45358	53439	48461

Method	Min	Max	Avg
Candidates + Random + Steepest + Edges (k=15)	45251	51868	48201
Candidates + Random + Steepest + Edges (k=20)	45356	51272	48294
LM Candidates + Random + Steepest + Edges (k=10)	46111	53213	49201
LM Candidates + Random + Steepest + Edges (k=20)	46324	53526	49391
<b>MSLS (200 iterations)</b>	<b>45799</b>	<b>45799</b>	<b>45799</b>
<b>ILS (time limit = 3246 ms)</b>	<b>43456</b>	<b>44291</b>	<b>43613</b>

**Key findings:**

- **ILS achieves the best average objective** on both instances among all methods tested
- ILS improves over the best construction heuristic by ~5% (TSPB) to ~1% (TSPA)
- MSLS also ranks among the top methods, competitive with best construction heuristics
- Both iterative methods significantly outperform simple local search variants

**TSPA Instance - Running Times (ms):**

Method	Min	Max	Avg
Random	0.0000	0.0005	0.0001
Nearest Neighbor (end only)	0.0342	0.0443	0.0386
Nearest Neighbor (any position)	1.4475	1.6102	1.4733
Greedy Cycle	2.6042	2.7999	2.6237
Greedy 2-Regret	2.5985	2.6897	2.6457
Greedy Weighted (2-Regret + BestDelta)	2.5949	2.6785	2.6246
Nearest Neighbor Any 2-Regret	1.4477	2.0777	1.5523
Nearest Neighbor Any Weighted (2-Regret + BestDelta)	1.4811	1.6774	1.5962
LS Random + Steepest + Nodes	19.4226	31.8627	24.3442
LS Random + Greedy + Nodes	1.9020	6.7854	3.5262
LS Random + Greedy + Edges	1.5978	4.0140	2.5544
LS Greedy + Steepest + Nodes	2.9704	6.4356	3.7700
LS Greedy + Steepest + Edges	3.0123	4.6928	3.5529
LS Greedy + Greedy + Nodes	3.3892	5.0970	3.8432
LS Greedy + Greedy + Edges	3.4178	4.9547	3.8581
LS Random + Steepest + Edges	14.4044	18.3997	16.2536
LM Random + Steepest + Edges	4.2332	8.1270	5.5978
Candidates + Random + Steepest + Edges (k=5)	3.7847	8.4170	4.4769

Method	Min	Max	Avg
Candidates + Random + Steepest + Edges (k=10)	5.3354	6.7890	6.0042
Candidates + Random + Steepest + Edges (k=15)	7.2444	9.2760	8.0441
Candidates + Random + Steepest + Edges (k=20)	8.9079	11.0741	9.9455
LM Candidates + Random + Steepest + Edges (k=10)	6.4400	8.3791	7.4459
LM Candidates + Random + Steepest + Edges (k=20)	18.9319	45.9974	22.4139
<b>MSLS (200 iterations)</b>	<b>3223.82</b>	<b>3482.74</b>	<b>3304.00</b>
<b>ILS (time limit = 3304 ms)</b>	<b>3304.07</b>	<b>3305.80</b>	<b>3304.60</b>

**TSPB Instance - Running Times (ms):**

Method	Min	Max	Avg
Random	0.0001	0.0008	0.0002
Nearest Neighbor (end only)	0.0352	0.1092	0.0461
Nearest Neighbor (any position)	1.4255	1.9735	1.4528
Greedy Cycle	2.5764	2.6887	2.6151
Greedy 2-Regret	2.5756	2.6728	2.6142
Greedy Weighted (2-Regret + BestDelta)	2.5693	2.7255	2.6273
Nearest Neighbor Any 2-Regret	1.4329	1.6191	1.5120
Nearest Neighbor Any Weighted (2-Regret + BestDelta)	1.4356	1.7028	1.5451
LS Random + Steepest + Nodes	19.4604	29.4551	23.6717
LS Random + Greedy + Nodes	2.1007	6.3568	3.6191
LS Random + Greedy + Edges	1.7279	3.0625	2.2735
LS Greedy + Steepest + Nodes	2.0857	5.8527	2.8188
LS Greedy + Steepest + Edges	2.0847	3.3959	2.5084
LS Greedy + Greedy + Nodes	2.1775	9.3875	2.8530
LS Greedy + Greedy + Edges	2.1311	3.6577	2.5706
LS Random + Steepest + Edges	14.1165	24.6737	16.5348
LM Random + Steepest + Edges	4.3736	7.0060	5.3759
Candidates + Random + Steepest + Edges (k=5)	4.1515	9.7641	4.6718
Candidates + Random + Steepest + Edges (k=10)	5.6526	9.0061	6.7328
Candidates + Random + Steepest + Edges (k=15)	7.8062	22.5641	9.0754
Candidates + Random + Steepest + Edges (k=20)	9.4218	12.0265	10.5970

Method	Min	Max	Avg
LM Candidates + Random + Steepest + Edges (k=10)	5.9539	8.3988	7.2321
LM Candidates + Random + Steepest + Edges (k=20)	19.2631	26.3872	22.4984
<b>MSLS (200 iterations)</b>	<b>3210.38</b>	<b>3381.45</b>	<b>3246.32</b>
<b>ILS (time limit = 3246 ms)</b>	<b>3246.40</b>	<b>3247.58</b>	<b>3246.79</b>

Key findings:

- MSLS and ILS are significantly slower than construction heuristics and single-run local search
- Both iterative methods take ~3.2-3.3 seconds per run
- **MSLS timing verification:** 200 iterations × 16.25 ms = 3250 ms (expected) vs 3304 ms (actual) = only **1.7% overhead**
- **ILS efficiency:** Performs ~3500 local search runs in the same time as MSLS's 200 runs (~17× more)
- The time investment is justified by the superior solution quality achieved

Visualizations

Best solutions found by MSLS and ILS visualized on both instances:

MSLS - TSPA	MSLS - TSPB
ILS - TSPA	ILS - TSPB

Analysis and Conclusions

Why ILS Outperforms MSLS

ILS achieves better results than MSLS for several reasons:

1. **More efficient exploration:** ILS performs ~17× more local search runs in the same time (3474 vs 200). This is because:
  - MSLS always starts from scratch (random solutions)
  - ILS starts from already-optimized solutions, making each LS run faster
2. **Guided search through perturbation:** ILS's perturbation strategy keeps the search in promising regions while still escaping local optima. The combination of multiple 2-opt moves and occasional node exchanges provides effective diversification.
3. **Accumulation of improvements:** ILS builds upon previous improvements, while MSLS treats each iteration independently.

MSLS Convergence Behavior

MSLS found the exact same solution (objective 70937 for TSPA, 45799 for TSPB) in all 20 runs:

- This suggests the search landscape has a dominant basin of attraction
- 200 random starts may be reaching the same local optimum repeatedly
- The local search from Assignment 3 is very effective at finding this optimum

## Perturbation Strategy Effectiveness

The ILS perturbation combining:

- **Multiple 2-opt moves** (2-5 depending on solution size): Changes tour structure significantly enough to escape local optima
- **Probabilistic node exchanges** (30% chance): Crucial for Selective TSP, allows exploring different node selections

This combination proved effective:

- Strong enough to escape the MSLS local optimum
- Gentle enough to stay in high-quality regions
- Results in consistently better solutions (2.3% to 4.8% improvement)

## Computational Efficiency

Both algorithms have very consistent runtimes:

- MSLS: 3304ms (TSPA), 3246ms (TSPB)
- ILS: 3304.6ms (TSPA), 3246.79ms (TSPB)
- Variance is minimal (< 1%), showing predictable performance

ILS achieves better results with the same computational budget by:

- Reusing optimized solutions as starting points
- Avoiding redundant exploration of the same basins

## Practical Recommendations

For the Selective TSP variant studied:

1. **ILS is the clear winner:** Better solution quality with same time budget
2. **Perturbation strength matters:** Our adaptive perturbation (scaling with solution size) proved effective
3. **MSLS shows diminishing returns:** 200 iterations may be excessive; many reach the same optimum
4. **Time-based stopping** (ILS) is more efficient than iteration-based (MSLS) for this problem

## Overall Conclusion

Iterated Local Search significantly outperforms Multiple Start Local Search on both test instances, achieving 2.3-4.8% better solutions in the same time. The key advantage is ILS's ability to efficiently explore the solution space by building upon previous improvements rather than starting from

scratch each time. The perturbation strategy successfully balances intensification (staying in good regions) and diversification (escaping local optima), particularly through the combination of structural changes (2-opt) and node selection changes (exchange moves).