College Tour Traveling Salesman Problem

Project Final Report

EMU427 Heuristic Methods for Optimization

ŞEVVAL NESİBE ARSLAN- 2220469053

ALI MNEIMNE- 2210469006

MEHMET ÖKSÜZOĞLU- 2220469036

MEHMET ARDA YENİÇULHA- 2220469061

HAKAN BERK YIKILMAZ- 2200469053

December 18, 2025 Hacettepe University

Contents

1. Introductıon

1.1. Background and Problem Improtance

The "College Tour Traveling Salesman Problem" (TSP) is a complex optimization challenge involving the coordination of visits to 58 universities across Istanbul. This problem is significantly more difficult than a standard TSP because it accounts for the unique geography of a megacity.

The importance of this problem lies in its constraints. Unlike theoretical models, real-world tours must account for:

Time Windows: Universities have specific visiting hours.

Physical Limitations: Travelers have daily limits on how many kilometers they can drive and how many hours they can spend on the road.

Prioritization: Certain locations are more critical than others and must be visited early in the schedule.

Solving this efficiently is important because inefficient routing leads to wasted time, higher fuel costs, and the inability to complete the tour within a set number of days.

1.2. Real Life Examples

While this project focuses on a college tour, the essence of this problam can be found in many important industries such as:

-E-commerce and Last-Mile Delivery: Companies like Amazon or DHL face the same "asymmetric and non-Euclidean road topology" of Istanbul. They must deliver packages within strict customer time windows while staying under daily driver work-hour limits.

-Medical Waste Collection: Logistics teams must visit hospitals to collect hazardous waste. If they miss a Time Window they cannot enter the facility, similar to the university visit constraints in our model.

-Drone Delivery Systems: Our approach to daily travel and range limits is modeled just like Vehicle Routing Problems with Drones, where flight distance is strictly limited by battery capacity.

-Technician Scheduling: Service companies (like internet providers or elevator repair teams) use these algorithms to ensure technicians spend more time fixing equipment and less time stuck in traffic between appointments.

2. Problem Formulatıon

2.1. Problem Introduction

The aim of this project is to devise an optimal and practical routing plan to visit each college during a college tour across Istanbul, where a number of individuals meet at Taksim to cover each one of its 58 universities. As a result, because the universities are scattered on both sides of Europe and Anatolia, this routing problem is actually an extension of the standard Traveling Salesman Problem. Every uinversity has a specific time window for visitors, and the vistors have limited traveling time and distance every day.

To make sure that this model is representative of such a journey's actual working conditions, a number of constraints are introduced into this model:

(A) Unique Visitation: The visitation to each university can at most occur once to avoid unnecessary duplication and to cover each of the 58 campuses comprehensively.

(B) Flow Balance: For each university visited, there is one incoming and one outgoing arc to assign, which ensures a continuous route.

(C) Time Windows: The visit to certain universities may occur within fixed time periods, which could represent visitation hours or other university activities.

(D) Daily Travel and Time Limitations: Since this group begins and ends every day at Taksim Square, they can travel no more than a designated amount each day due to the constraints of traveling within the city.

(E) Subtour Elimination (MTZ): Miller-Tucker-Zemlin constraints are introduced to forbid any disconnected subtours, thus making sure that there is one single circuit which traverses all universities.

(F) Prioritization: Some universities are identified as priority nodes, such as those having fixed meeting schedules, which need to be accessed early or within specific windows.

2.1.1. LP Formulation

The following mathmatical model describes our Queen of College Tours TSP problem. It includes all the distance and time constraints, the time window constraint, and it insures that the prioroty universities are visited first.

Sets:

- $V = \{0,1,\dots,n\}$: {0,1,…,n}: nodes; 0 = depot (home/start), 1..n = universities.

  (This is the complete set of nodes in the problem. 0 is the start and ending point everyday)

- $D = \{\{1,\dots,d\_max\}$: days (Represents the total number of days the tour lasts.Each day is the group starts and ends at the depot)

- $P \subseteq \{1, \dots, n\}$: index set of priority universities. (This is a subset of universities that are marked as important, we have this set because we have a Prioritization constraint later)

Parameters

- dis_{ij}: distance between node i and node j (Taken from the distance Excel matrix, represents real driving distance between universities)
- t_{ij}: actual travel time from node i to node j (Represents actual travel duration in minutes)
- a_i, b_i: time window (A visit to university i must happen between these bounds. If a university is visited, arrival time must respect the window.)
- $s_{s,d}$, $s_{e,d}$: daily start and end time for each day d (Defines the working hours of each day. The depot's arrival time must be within this interval.)
- R_d^{dis}: maximum allowed total distance for day d (meters)
- R_d^{t}: maximum allowed total travel time for day d (minutes)
- W_i: priority weight (Represents how important university i is)
- M: big-M constant
- d: day

Decision Variables

- $x_{ij}^d \in \{0,1\}$: equals 1 if edge i–j is used on day d; i,j ∈ V.
- $y_i^d \in \{0,1\}$: equals 1 if university i is visited on day d (i ≥ 1).
- $t_i^d \geq 0$: arrival time at node i on day d
- $u_i^d \geq 0$: MTZ sequencing variable (Used for subtour elimination. Ensures that each day's route forms a single connected loop and eliminates cycles that do not include the depot.)

Objective function

$$\min \sum_{d \in D} \sum_{i \in V} \sum_{j \in V} dis_{ij}\, x_{ij}^d \; - \; \lambda \sum_{i \in P} \sum_{d \in D} W_i \cdot (D - d)\, y_i^d$$

min Total Travel Distance - Priority/Early Visit Reward

Total Travel Distance sums up the distances of the arcs we used. It is to be minimized

dis_{ij}: distance between node i and node j

$x_{ij}^{d} \in \{0,1\}$ shows if edge ij is used on day i

Priority/Early Visit Reward encourages visiting important (priority) universities as early as possible, it is to be maximized.

Wi is the prioroty weight of the university i. D subtracted by d shows how early we visited university i (for example if our total days are 10 and we visited the university at day 3, D-d equals 7)

$y_{i}^{d} \in \{0,1\}$ shows if university i is visted on day d

Constraints

A. Each university is visited exactly once

$$\sum_{d \in D} y_i^d = 1 \qquad \forall i \in \{1 \dots n\}.$$

B. Daily flow balance (each visit has one incoming and one outgoing arc)

$$\sum_{i \in V} x_{ij}^d = y_j^d \quad \forall j \in V \setminus \{0\}, \forall d \in D.$$

$$\sum_{j \in V} x_{ij}^d = y_i^d \quad \forall i \in V \setminus \{0\}, \forall d \in D.$$

$$\sum_{j \in V \setminus \{0\}} x_{0j}^d = 1 \quad \forall d \in D,$$

(The daily route must start from the depo and must leave the depo once)

$$\sum_{i \in V \setminus \{0\}} x_{i0}^d = 1 \quad \forall d \in D.$$

(The daily route must end at the depo and must return to the depo once)

C.Time Window

Linking constraint:

$$t_j^d \geq t_i^d + time_{ij} - M \cdot (1 - x_{ij}^d) \forall i, j \in V, \forall d \in D.$$

If the route goes from i to j on day d, then the arrival time at j must be equal to the arrival time at i plus the travel time.
If the route does not use arc ij, the big-M term deactivates the constraint.
This ensures logical time progression along the route.

Time Window:

$$a_i \leq t_i^d \leq b_i + M(1 - y_i^d) \forall i \in \{1..n\}, \forall d \in D.$$

If university i is visited on day d, the arrival time must fall inside its allowed interval [ai, bi]. If it is not visited, the big-M term relaxes the interval. Thus models available visiting periods for the universities.

Start-time control:

$$t_0^d = s_{s,d}$$

$$t_0^d \leq s_{e,d}$$

The depot (node 0) has a fixed start time for each day. The start time must also be before the day's allowed ending time. This establishes a consistent daily departure schedule.

End of day:

$$t_0^{d,return} \leq s_{e,d}.$$

The return to the depot must happen before the day's ending time.

D. Limited travel range and time

$$\sum_{i \in V} \sum_{j \in V} dis_{ij}\, x_{ij}^d \leq R_d^{dis} \quad \forall d \in D\ (distance)$$

$$\sum_{i \in V} \sum_{j \in V} t_{ij}\, x_{ij}^d \leq R_d^t \quad \forall d \in D\ (time)$$

The total driving distance traveled on day d cannot exceed the daily distance limit, The total driving time on day d also cannot exceed the daily time limit.

E. Subtour elimination (MTZ)

$$u_i^d - u_j^d + (n+1)\, x_{ij}^d \le n \quad \forall i \ne j,\ i,j \in V \setminus \{0\},\ \forall d \in D.$$

$$1 \le u_i^d \le n \cdot y_i^d \quad \forall i \in V \setminus \{0\},\ \forall d \in D.$$

The variables u_i^d represent the visiting order on day d, and this constraint forces them to be on a single continuous route. (It eliminates cycles that do not include the depot)

F. Prioritization

A constraint ensuring at least k_p of the priority universities are visited within the first D_0 days (e.g., first 5 days):

$$\sum_{i \in P} \sum_{d=1}^{D_0} y_i^d \ge k_p$$

3. Description of the Metaheuristic Algorithm

3.1. The Algorithm

To solve the Istanbul College Tour TSP, we adopt the **Adaptive Large Neighborhood Search (ALNS)** metaheuristic. This choice is motivated by the fact that the problem extends the classical TSP with multiple real-world constraints (multi-day routing, time windows, daily distance/time limits, and prioritization), making exact optimization approaches computationally impractical at a 58-node scale.

At a high level, ALNS iteratively improves a feasible multi-day schedule through a **Destroy–Repair** cycle. In each iteration, a subset of visited universities is removed from the current solution (destroy) and then reinserted using a constructive heuristic that explicitly checks feasibility with respect to daily budgets and time-window consistency (repair). Since such constrained search spaces are prone to premature convergence, we integrate a **Simulated Annealing** acceptance rule that can occasionally accept non-improving moves, enabling the method to escape local minima. Finally, after each repair step, a lightweight **intra-day 2-opt** local search is applied to refine route ordering within each day without changing the day assignments.

Why ALNS as Core Algorithm?

Due to the scale of the problem and the presence of multiple interacting constraints, exact optimization approaches were found to be computationally impractical. In addition, relying solely on a classical local search method such as 2-opt was also deemed insufficient for this problem.

The 2-opt algorithm is a well-known local improvement heuristic that operates by swapping two edges within an existing route to reduce total travel distance. While 2-opt is effective for refining the ordering of visits within a single route, it is inherently limited to a very small neighborhood structure**.** As a result, it can only perform incremental improvements and is highly prone to becoming trapped in local optima. More importantly, 2-opt cannot modify the assignment of universities across different days, which constitutes the core complexity of the multi-day College Tour problem.

In this study, the main challenge is not only the sequencing of visits within a day, but also the **global distribution of universities across** multiple days under time, distance, and prioritization constraints. Addressing such large-scale structural decisions requires a mechanism capable of performing substantial modifications to the solution, which exceeds the capabilities of standalone local search methods.

For this reason, the **Adaptive Large Neighborhood Search (ALNS)** framework was adopted as core optimization strategy. ALNS enables the algorithm to escape local minima by deliberately destroying part of the current solution and rebuilding it in a different way. This allows universities to be reassigned between days, leading to fundamentally different route structures that cannot be reached through local edge exchanges alone.

To further enhance the search process, **Simulated Annealing (SA)** was integrated as the acceptance criterion. The SA mechanism allows the temporary acceptance of worse solutions with a controlled probability, particularly during early stages of the search. This feature is crucial for preventing premature convergence and for promoting exploration in a highly constrained and rugged solution space.

Finally, **2-opt local search** was retained as a complementary component within the ALNS framework. Rather than serving as the main optimization engine, 2-opt is applied after the repair phase to refine intra-day routes. In this hybrid structure, ALNS is responsible for global exploration and structural changes**,** Simulated Annealing governs solution acceptance and diversification**,** and 2-opt provides local exploitation and fine-tuning.

This combination allows the algorithm to effectively balance exploration and exploitation, making it well-suited for solving large-scale, multi-day routing problems with complex real-world constraints such as those present in the Istanbul College Tour scenario.

The Pseudo Code:

The following pseudocode presents the integrated logical flow of the proposed ALNS metaheuristic combined with 2-opt & Simulated Annealing. The algorithm takes the sets of nodes (*V*) days (*D*), and priority universities (*P*) along with the calculated travel-time and distance matrices, as input parameters. It initializes a constructive feasible solution and enters an iterative improvement loop governed by the adaptive operator selection and the Simulated Annealing acceptance criterion described in Sections 5.3 and 5.4. The procedure terminates when the maximum time limit or iteration count is reached, returning the best-found schedule (S_{best}).

```
Algorithm: ALNS for College Tour TSP

Input:
    V(Nodes), D(Days), P(Priority Set)
    Matrices: dis[i][j], time[i][j]
    Params: TimeWindows[a, b], DailyLimits R_dis, R_time


Output: Best Solution S * (Set of routes for all days)

1.INITIALIZATION
 S_current ← ConstructInitialSolution()
 S_best ← S_current
 Assign weights w_destroy, w_repair to operators (based on [4])
 T ← Initial Temperature

2.  WHILE (Time < TimeLimit AND Iterations < MaxIter) DO:

3.      // --- SELECTION PHASE (Adaptive Mechanism) ---
        // Select operators based on past performance, logic from [3]
        Op_destroy ← SelectRoulette(w_destroy)
        Op_repair  ← SelectRoulette(w_repair)
        q ← Determine removal size

4.      // --- DESTROY PHASE ---
        // Logic similar to [2] for node removal
        S_temp, RequestBank ← Apply(Op_destroy, S_current)

5.      // --- REPAIR PHASE ---
        For each node u in RequestBank:
            BestPos ← Null

            For each Day d in D:
                For each Position k in Route[d]:

                    // FEASIBILITY CHECKS (Modeled after [1])
                    TentativeRoute ← Insert(u, d, k)
                    Calculate Arrival Times (t_i)

                    If(t_i <= b_i) AND                    // Time Window
                       (TotalDist <= R_dis[d]) AND         // Daily Distance Limit
                       (TotalTime <= R_time[d]) THEN:      // Daily Time Limit

DeltaCost ← CalculateCostChange(TentativeRoute)
                       If DeltaCost<BestCost:
                           BestPos ← (d, k)

            ApplyInsertion(u, BestPos)

6.      // --- LOCAL SEARCH & ACCEPTANCE ---
        S_new ← Apply 2 - Opt(S_new)

        // Simulated Annealing acceptance criterion
        DeltaE ← Cost(S_new) - Cost(S_current)
        If DeltaE< 0 OR Random(0,1) < exp(-DeltaE / T):
            S_current ← S_new
            UpdateWeights(Outcome) // Update logic based on solution quality

    Update Temperature T

7.  END WHILE

8.  RETURN S_best
```

3.2 The Implementation

Our implementation focuses on how the ALNS handles the unique constraints of the Istanbul "College Tour" rather than standard data processing.

3.2.1. Real-World Distance Foundation

To ensure the algorithm makes realistic decisions, we calculated a distance matrix using the Open Source Routing Machine (OSRM). Unlike simple straight lines, this gives the algorithm the actual driving distances and times across Istanbul's bridges and one-way streets. This matrix serves as the base data that all our operators use to evaluate costs.

3.2.2. Learning Mechanism (Adaptive Weights)

The adaptive component of the ALNS framework is a key feature that distinguishes it from standard large neighborhood search methods. Instead of selecting destroy and repair operators uniformly at random, the algorithm dynamically adjusts their selection probabilities based on past performance.

Each destroy and repair operator is associated with a weight that reflects its historical contribution to solution quality. During the search, operators are selected using a roulette-wheel mechanism proportional to their current weights.

After each iteration, the weights of the selected operators are updated according to the outcome of the move:

- If the new solution improves the **global best solution**, the corresponding operators receive a high reward.
- If the new solution improves the **current solution** but does not outperform the global best, a moderate reward is assigned.
- If the new solution is accepted due to the **Simulated Annealing criterion** despite being worse, a small reward is given to encourage exploration.

This reward-based update mechanism allows the algorithm to gradually learn which destroy and repair strategies are most effective under Istanbul's non-Euclidean road structure and tight operational constraints. Over time, operators that consistently produce high-quality or useful exploratory moves are selected more frequently, leading to a self-adaptive balance between diversification and intensification.

3.2.3. Procedural Constraint Management

Instead of allowing the algorithm to create infeasible routes and fixing them later, all hard constraints are handled directly during the repair phase. This ensures that every solution considered by the algorithm is feasible.

When the algorithm tries to insert a university into a daily route, it immediately checks whether this insertion respects the daily travel limits and the university's time window. The total travel time and distance of the day are updated step by step during this process.

If adding a university causes the daily route to exceed the 8-hour limit or violates a time window, the insertion is rejected right away. The algorithm then tries alternative positions or assigns the university to a different day. In this way, infeasible routes are never accepted or carried forward.

By embedding constraint checks into the repair procedure, the search is limited to feasible solutions only. This reduces unnecessary evaluations and helps the algorithm converge more smoothly, which is especially important for routing problems with strict operational limits.

Hard constraints enforced during the repair phase include:

- Each university must be visited exactly once during the entire tour.
- Each daily route must start and end at the depot (Taksim)**.**
- The total travel time of a day must not exceed 8 hours.
- University-specific time windows must be respected.
- No university can be visited more than once within the same day.

Soft constraints addressed in the objective function include:

- Visiting high-priority universities earlier in the tour.
- Balancing daily workloads across the planning horizon.
- Minimizing total travel distance beyond feasibility requirements.

3.2.4. Priority-Based Repair

To handle the prioritization constraint, a dedicated repair operator is used during the ALNS process. This operator orders the universities in the request bank according to their predefined importance weights.

Universities with higher priority are inserted first and are preferentially assigned to earlier days of the 20-day tour. This increases the likelihood that important campuses are visited as early as possible.

By guiding the insertion order in this way, the repair phase directly supports the objective function, which rewards early visits to high-priority universities. As a result, prioritization is enforced procedurally rather than through heavy penalty terms.

3.2.5. Github link

The codes for our work are in the following github repository.

https://github.com/sevvaln/college-tour-tsp

4. Experimental Results and Evaluation

This chapter presents the experimental results of the Adaptive Large Neighborhood Search (ALNS) algorithm developed for the College Tour Traveling Salesperson Problem (TSP) with constraints. It details the parameter tuning process, the final performance evaluation against a benchmark, and a detailed analysis of the best-found solution.

All experiments were conducted under the same computational environment to ensure consistency and statistical reliability. The experiments were performed on a Monster Abra A5 v19.1 laptop equipped with an Intel Core i5-12500H processor, 16 GB RAM, and a dedicated NVIDIA GPU with 4 GB VRAM, running the Windows operating system.

In this study, three main types of code were developed and used to evaluate the proposed Adaptive Large Neighborhood Search (ALNS) algorithm from different perspectives. Each code serves a distinct experimental purpose: (i) solving the base College Tour problem, (ii) tuning critical algorithm parameters, and (iii) testing robustness and scalability through stress testing.

4.1 Main ALNS Execution Code (Baseline Optimization)

The main execution code runs the ALNS algorithm on the full Istanbul College Tour instance using a fixed parameter configuration obtained from the tuning phase. The algorithm starts by loading the OSRM-based distance matrix, which represents realistic driving distances between 58 universities and the depot (Taksim).

After initializing a feasible solution, the algorithm iteratively improves the routing plan through the destroy–repair mechanism combined with Simulated Annealing. During execution, intermediate progress is logged at regular iteration intervals, reporting the current temperature, the best solution cost found so far, and feasibility status. This allows monitoring of convergence behavior throughout the optimization process.

The algorithm consistently maintains feasibility, ensuring that:

- each university is visited exactly once,
- daily routes start and end at the depot,
- time windows and daily travel limits are respected,
- and prioritization constraints are satisfied.

At the end of the run, the algorithm outputs a complete **20-day tour schedule**, where each day contains a feasible sequence of universities to be visited. In addition to the final routing plan, the execution reports:

- the **best total cost** achieved,
- the **overall feasibility status**,
- the **total runtime**,
- and a detailed **cost breakdown** summarizing the objective value.

For reproducibility and reporting purposes, the final solution is automatically exported to an Excel file, which includes the daily routes, cost information, and feasibility indicators. This execution represents the core optimization result of the study and serves as the baseline solution for all subsequent analyses, including parameter tuning comparisons and robustness evaluations.

4.2 Parameter Tuning Study

A comprehensive parameter tuning study was conducted to **improve the performance** of the ALNS algorithm, specifically focusing on the Simulated Annealing acceptance criterion components: the **Initial Temperature (Tinitial)** and the **Cooling Rate (**α).

4.2.1 Tuning Parameters and Criteria

The ranges for the parameters were determined based on preliminary runs and empirical observations grounded in literature regarding ALNS and Simulated Annealing applications[3]:

- **Initial Temperature:** The range tested was [300, 3000]. A high initial temperature was chosen to enable the algorithm to widely **explore** the search space. Lower T values represent a faster local search or **exploitation** phase.

- **Cooling Rate:** The range tested was [0.95, 0.99]. Keeping α close to 1 aims to prevent premature convergence to local optima by maintaining the Metropolis criterion's probability of accepting worse solutions for an extended period. This interval was based on the assumption that a slow cooling schedule is more suitable for such complex constrained problems

While the ALNS framework contains several algorithmic parameters such as destroy–repair operator weights, removal size, and local search configurations, preliminary experiments indicated that these parameters had a secondary influence compared to the Simulated Annealing acceptance criterion.

Therefore, these parameters were fixed at empirically reasonable values to reduce the dimensionality of the tuning process. The tuning effort was deliberately concentrated on the Initial Temperature and Cooling Rate, as these parameters directly control the balance between diversification and intensification and were observed to have the most significant impact on convergence behavior and solution quality.

Each parameter configuration was evaluated using multiple independent runs to mitigate the effect of randomness inherent in the algorithm. The primary evaluation metric was the **Mean Final Best Cost** obtained for each parameter combination.

- **Comprehensive Evaluation:** In addition to the mean cost, the consistency of the solution quality and the algorithm's efficiency (Mean Runtime) were also considered to ensure a **cost-efficiency trade-off**. The goal was to select the configuration that offered low mean cost, low standard deviation, and a reasonable runtime.

The following table summarizes the parameter values tested in the tuning study:

Tablo 1. Algorithm Tuning Configurations

Parameter	Possible values	Test values
Start temperature (T_{start})	$T_{start} > 0$	{300,1000,3000}
Temperature decrease (α)	0<α<1	{0.95,0.98,0.99}
Finish temperature (T_{finish})	$T_{finish} \geq 0$	$\left[9.67 \times 10^{-4}, \ldots, 9.99 \times 10^{-4}\right]$

The final temperature was not treated as an independent tuning parameter. Instead, it was implicitly determined by the initial temperature, cooling rate, and the number of iterations as:

$$T_{finish} = T_{start} \cdot \alpha^{N}$$

The final temperatures reached across the different parameter configurations are reported for completeness and transparency, providing insight into the effective cooling behavior of the algorithm.

INITIAL_T	ALPHA_T	Final Temperature	Final Temperature
300	0.99	1255	9.98×10^{-4}
1000	0.99	1375	9.96×10^{-4}
3000	0.99	1484	9.99×10^{-4}
300	0.98	625	9.85×10^{-4}
1000	0.98	684	9.97×10^{-4}
3000	0.98	739	9.84×10^{-4}
300	0.95	246	9.93×10^{-4}
1000	0.95	270	9.67×10^{-4}
3000	0.95	291	9.88×10^{-4}

4.2.2 Tuning Results and Selection

The tuning results were summarized using a heatmap representation , demonstrating the average final best cost for each parameter combination.

Tablo 2. Algorithm Tuning Metrics

INITIAL_T	**ALPHA_T**	**Mean Final Cost**	**Min Cost**	**Std Dev**	**Feasible Rate**	**Mean Runtime (s)**
300	0.99	1,271,052	1,252,955	11,890	1.00	236
1000	0.99	1,272,532	1,256,399	11,697	1.00	254
3000	0.99	1,273,294	1,243,297	14,980	1.00	346
3000	0.98	1,282,335	1,246,212	17,374	1.00	138
300	0.98	1,282,796	1,263,257	13,974	1.00	120
1000	0.98	1,283,660	1,260,190	14,954	1.00	127
1000	0.95	1,303,580	1,287,370	11,635	1.00	50
300	0.95	1,303,766	1,290,379	10,757	1.00	68
3000	0.95	1,312,372	1,284,187	19,906	1.00	54

Selected Configuration: Based on the tuning results, the parameter combination **T_initial = 300 and alpha = 0.99** achieved the lowest average final cost without introducing unnecessary computational overhead. Therefore, this configuration was selected for the final experiments.

4.3. Final Algorithm Performance Evaluation

After selecting the best parameter configuration , the final performance of the algorithm was evaluated using a **Cold-Start** initialization strategy. In this setting, the initial solution is generated randomly by shuffling non-depot nodes and distributing them across daily routes. This approach avoids bias and ensures a fair evaluation of the algorithm. The selected configuration was executed over 10 independent runs, each with a different random seed.

4.3.1 Aggregate Run Statistics

The performance of the algorithm is reported using summary statistics to provide a representative evaluation.

Tablo 3. Cold Start Summary Statistics

Metric	Value
Number of runs	10
Mean Final Cost	1,277,486.62
Standard Deviation	12,190.02
Best Final Cost	1,255,127.24
Worst Final Cost	1,292,663.90
Average Runtime (sec)	294.37

The results indicate that the algorithm produces consistent solutions across different runs. The relatively small variation in final costs demonstrates the robustness of the selected parameter configuration under cold-start conditions.

4.3.2 Improvement Analysis Against a Random Baseline

To quantify the effectiveness of the ALNS approach, the final result was compared against the average Initial Cost obtained via random route assignment. This serves as the zero point before the adaptive mechanisms of ALNS are engaged.

Tablo 4. Optimization Baseline and Improvement

Mean Initial Cost	2,308,901.91
Minimum Initial Cost	2,223,235.46
Maximum Initial Cost	2,421,774.30
Initial Cost Std. Dev.	66,280.64
ALNS Best Final Cost	1,255,127.24
ALNS Improvement Percentage	%45.64

The random assignment yielded a high average cost of 2,308,901.91 m and high instability with a standard deviation of 66,280.64 m. The applied ALNS meta-heuristic achieved **45.64% improvement** in the final best solution compared to this random baseline.

4.3.3 Consistency Analysis Against the Best-Found Solution (Gap)

The gap is calculated between the Mean Final Cost and the Best Final Cost found across the 10 independent runs. Since no known optimal solution or analytical lower bound exists for this real-world problem instance, the best-found solution across all independent runs is used as a reference point. This practice is common in heuristic studies on custom datasets and allows the evaluation of algorithmic consistency rather than absolute optimality.

$$Gap = 100\ .\ \frac{(1{,}277{,}486.62 - 1{,}255{,}127.24)}{1{,}255{,}127.24} \approx 1.78\%$$

The resulting gap therefore represents the average deviation from the best solution discovered during the experimental campaign, providing a meaningful measure of robustness and stability under stochastic initialization.

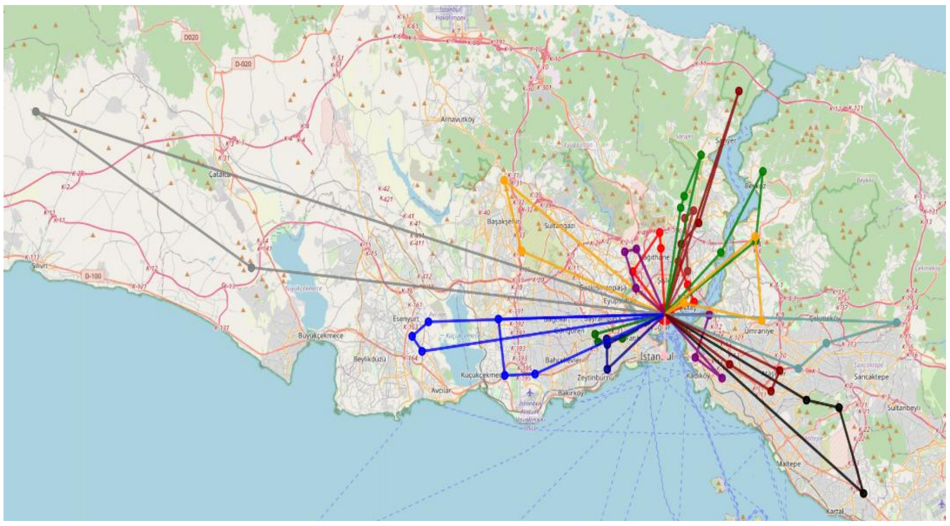
4.3.4 Local Optimality Validation (Warm Start Experiment)

To further validate the quality and local optimality of the best-found solution, an additional experiment was conducted using a **Warm Start strategy**. In this approach, the best solution obtained from the Cold Start runs was used as the initial solution for a new set of ALNS executions. The primary objective was to determine if the algorithm's intensification mechanisms could refine the current solution further or if the solution had converged to a strong local optimum. Results from this warm start showed that the ALNS failed to achieve any **significant improvement** upon the initial seed. This lack of further enhancement strongly suggests that the best-found solution resides in a **deep and high-quality local optimum** within the search space, and the current set of neighborhood operators or the cooling schedule lacks the necessary perturbation power to escape this basin and find a globally superior solution.

4.4 Detailed Analysis of the Best Solution

The visualization highlights that the algorithm naturally groups geographically close universities within the same day, reducing unnecessary cross-city travel. Routes are generally compact and avoid excessive back-and-forth movements, indicating effective exploitation of the road-network-based distance matrix. In addition, the number of universities per day remains balanced, demonstrating that daily travel limits and time window constraints are respected. The radial structure around the depot further confirms that all routes consistently start and end at Taksim, as required by the problem formulation.

Tablo 5. Best Solution Routes Visualization



The figure illustrates the final 20-day routing plan generated by the ALNS algorithm. Each color represents a daily route starting and ending at the depot (Taksim), clearly showing the geographical clustering of universities and the balanced distribution of visits across days.

4.4.1. Daily Route Distribution and Constraint Management

The best solution's daily route schedule and constraint adherence are detailed below.

Tablo 6. Best Solution Route Schedule

Day	Route	Number of Universities Visited	Total Route Distance (km)	Total Travel Time (hour)	Number of Priority Universities
1	59-41-58-13-59	3	18.97	6.63	3
2	59-26-23-42-59	3	73.47	8.45	1
3	59-55-30-59	3	54.98	7.83	3
4	59-56-54-48-59	3	35.49	7.18	1
5	59-2-16-59	2	59.00	5.97	0
6	59-50-38-46-59	3	54.99	7.83	3
7	59-51-10-3-59	3	73.66	8.46	1
8	59-6-43-44-59	3	21.88	6.73	0
9	59-32-53-22-59	3	118.16	9.94	1
10	59-35-45-24-59	3	16.53	6.55	0
11	59-18-33-59	2	190.58	10.35	0
12	59-57-47-25-59	3	78.17	8.61	0
13	59-9-49-37-59	3	26.93	6.90	0
14	59-11-7-40-59	3	21.95	6.73	0
15	59-27-19-39-59	3	21.40	6.71	0
16	59-34-20-28-59	3	53.93	7.80	0
17	59-21-31-15-59	3	37.94	7.26	0
18	59-14-22-36-59	3	23.71	6.79	0
19	59-4-17-5-59	3	44.09	7.47	0
20	59-29-1-12-59	3	43.79	7.47	0
Total	-	58	1069.61	151.65	13

Tablo 7. Universities

Universities	node_id
Acıbadem Mehmet Ali Aydınlar Üniversitesi	1
Altınbaş Üniversitesi	2
ATAŞEHİR ADIGÜZEL MESLEK YÜKSEKOKULU	3
BAHÇEŞEHİR ÜNİVERSİTESİ	4
BEYKOZ ÜNİVERSİTESİ	5
BEZM-İ ÂLEM VAKIF ÜNİVERSİTESİ	6
BİRUNİ ÜNİVERSİTESİ	7
BOĞAZİÇİ ÜNİVERSİTESİ	8
DEMİROĞLU BİLİM ÜNİVERSİTESİ	9
DOĞUŞ ÜNİVERSİTESİ	10

FATİH SULTAN MEHMET VAKIF ÜNİVERSİTESİ	11
FENERBAHÇE ÜNİVERSİTESİ	12
GALATASARAY ÜNİVERSİTESİ	13
HALİÇ ÜNİVERSİTESİ	14
IŞIK ÜNİVERSİTESİ	15
İBN HALDUN ÜNİVERSİTESİ	16
İSTANBUL 29 MAYIS ÜNİVERSİTESİ	17
İSTANBUL AREL ÜNİVERSİTESİ	18
İSTANBUL ATLAS ÜNİVERSİTESİ	19
İSTANBUL AYDIN ÜNİVERSİTESİ	20
İSTANBUL BEYKENT ÜNİVERSİTESİ	21
İSTANBUL BİLGİ ÜNİVERSİTESİ	22
İSTANBUL ESENYURT ÜNİVERSİTESİ	23
İSTANBUL GALATA ÜNİVERSİTESİ	24
İSTANBUL GEDİK ÜNİVERSİTESİ	25
İSTANBUL GELİŞİM ÜNİVERSİTESİ	26
İSTANBUL KENT ÜNİVERSİTESİ	27
İSTANBUL KÜLTÜR ÜNİVERSİTESİ	28
İSTANBUL MEDENİYET ÜNİVERSİTESİ	29
İSTANBUL MEDİPOL ÜNİVERSİTESİ	30
İSTANBUL NİŞANTAŞI ÜNİVERSİTESİ	31
İSTANBUL OKAN ÜNİVERSİTESİ	32
İSTANBUL RUMELİ ÜNİVERSİTESİ	33
İSTANBUL SABAHATTİN ZAİM ÜNİVERSİTESİ	34
İSTANBUL SAĞLIK VE SOSYAL BİLİMLER MESLEK YÜKSEKOKULU	35
İSTANBUL SAĞLIK VE TEKNOLOJİ ÜNİVERSİTESİ	36
İSTANBUL ŞİŞLİ MESLEK YÜKSEKOKULU	37
İSTANBUL TEKNİK ÜNİVERSİTESİ	38
İSTANBUL TİCARET ÜNİVERSİTESİ	39
İSTANBUL TOPKAPI ÜNİVERSİTESİ	40
İSTANBUL ÜNİVERSİTESİ	41
İSTANBUL ÜNİVERSİTESİ-CERRAHPAŞA	42
İSTANBUL YENİ YÜZYIL ÜNİVERSİTESİ	43
İSTİNYE ÜNİVERSİTESİ	44
KADİR HAS ÜNİVERSİTESİ	45
KOÇ ÜNİVERSİTESİ	46
MALTEPE ÜNİVERSİTESİ	47
MARMARA ÜNİVERSİTESİ	48
MEF ÜNİVERSİTESİ	49
MİMAR SİNAN GÜZEL SANATLAR ÜNİVERSİTESİ	50
ÖZYEĞİN ÜNİVERSİTESİ	51
PİRİ REİS ÜNİVERSİTESİ	52
SABANCI ÜNİVERSİTESİ	53
SAĞLIK BİLİMLERİ ÜNİVERSİTESİ	54

TÜRK-ALMAN ÜNİVERSİTESİ	55
ÜSKÜDAR ÜNİVERSİTESİ	56
YEDİTEPE ÜNİVERSİTESİ	57
YILDIZ TEKNİK ÜNİVERSİTESİ	58
Taksim Meydan	59

The core constraints, such as the Daily Duration Limit (8 hours) and Time Windows, were treated as **Soft Constraints** within the algorithm[63]. This approach penalizes constraint violations by adding an equivalent cost to the objective function, instead of strictly excluding infeasible solutions.

- **Rationale for Constraint Violations (Daily Time Overruns):** The observed overruns of the 8-hour daily limit on Days 9, 11, and 12 are a consequence of the designed feature, not an error.
- **Optimization Decision:** The algorithm seeks to find the **minimum total cost**. In some cases, the **penalty cost** incurred by violating a constraint is lower than the **additional travel distance cost** required to strictly adhere to it. This demonstrates the flexibility and real-world adaptability of the ALNS.

The difference between the reported Final Cost and the calculated Total Route Distance is precisely explained by the structure of the objective function. This difference corresponds to the Total Penalty Cost accumulated during the solution process:

Total Penalty Cost = Reported Cost – Calculated Route Distance

Total Penalty Cost = 1,255,274.24 m – 1,069,610.00 m $\approx$ 185,517.24m

This 185,517.24 m penalty cost primarily consists of the sum of Daily Duration Overrun Penalties, Priority Violation Penalties, and Time Window Delay Penalties.

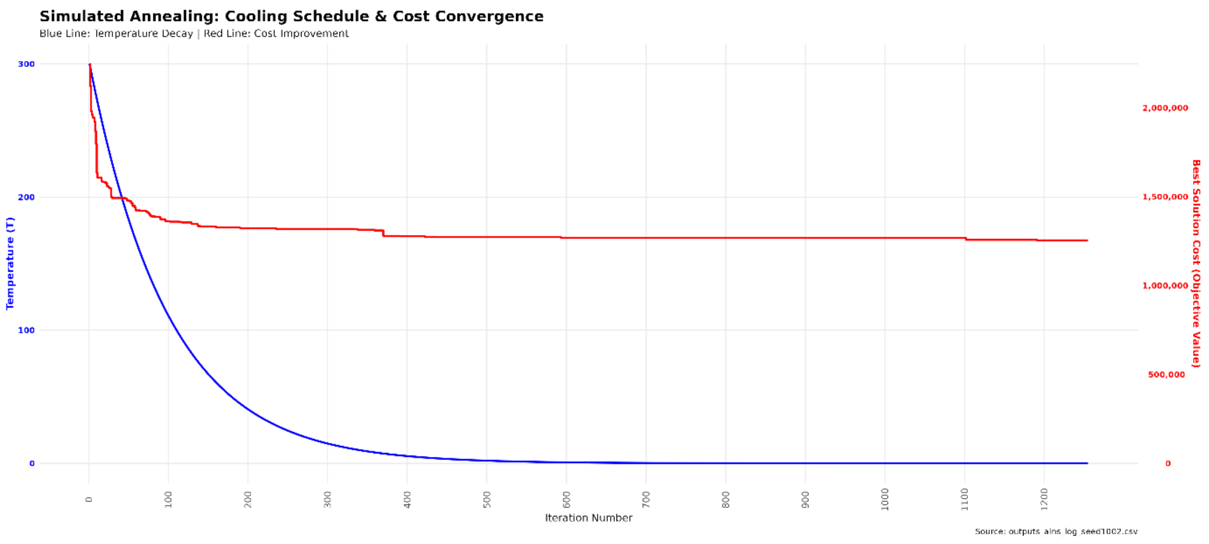
4.4.2 ALNS Algorithm Performance and Parameter Tuning Analysis

This section presents a graphical analysis of the performance of the Adaptive Large Neighborhood Search (ALNS) algorithm and the impact of key Simulated Annealing parameters. Using convergence plots, interaction graphs, heatmaps, and boxplots, we analyze how the algorithm evolves over iterations, how parameter choices affect solution quality, and how stable the results are across repeated runs. These visualizations provide insight into both the efficiency of the search process and the robustness of the selected parameter configuration.

The experimental analysis and visualization components of this study were implemented in R, while the core ALNS algorithm and auxiliary optimization codes were implemented in Python. This choice was motivated by R’s strong ecosystem for statistical evaluation and its advanced visualization libraries, which allow clearer representation of parameter interactions and result variability across multiple runs.

1. Algorithm Convergence Behavior

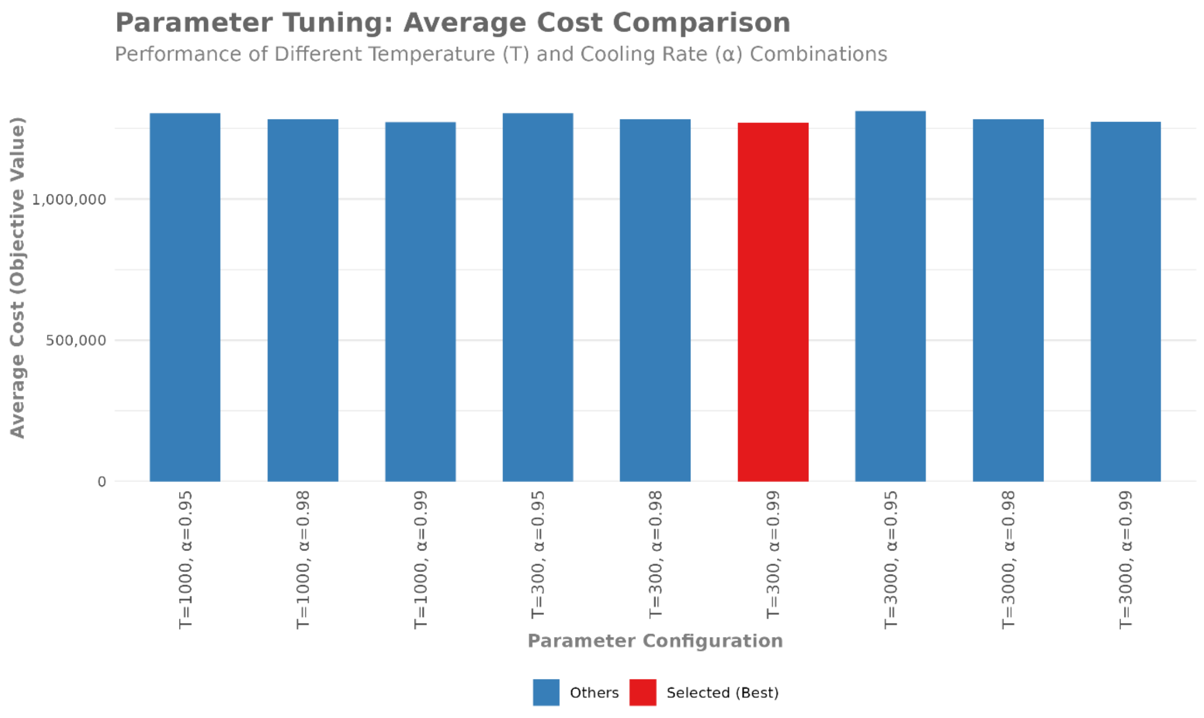
The convergence behavior of the Adaptive Large Neighborhood Search (ALNS) algorithm demonstrates a classic optimization trajectory. The initial iterations show a steep descent in the objective function value (Total Cost), indicating that the algorithm effectively improves the solution quality rapidly in the early stages. As the iterations proceed, the curve flattens, representing the exploitation phase where the algorithm refines the solution.The integration of Simulated Annealing (SA) is clearly visualized in the cooling schedule plot. The blue line represents the exponential decay of the temperature (*T*), while the red line tracks the best solution cost. The high initial temperature allows for the acceptance of worse solutions to escape local optima, which stabilizes as the temperature approaches zero.

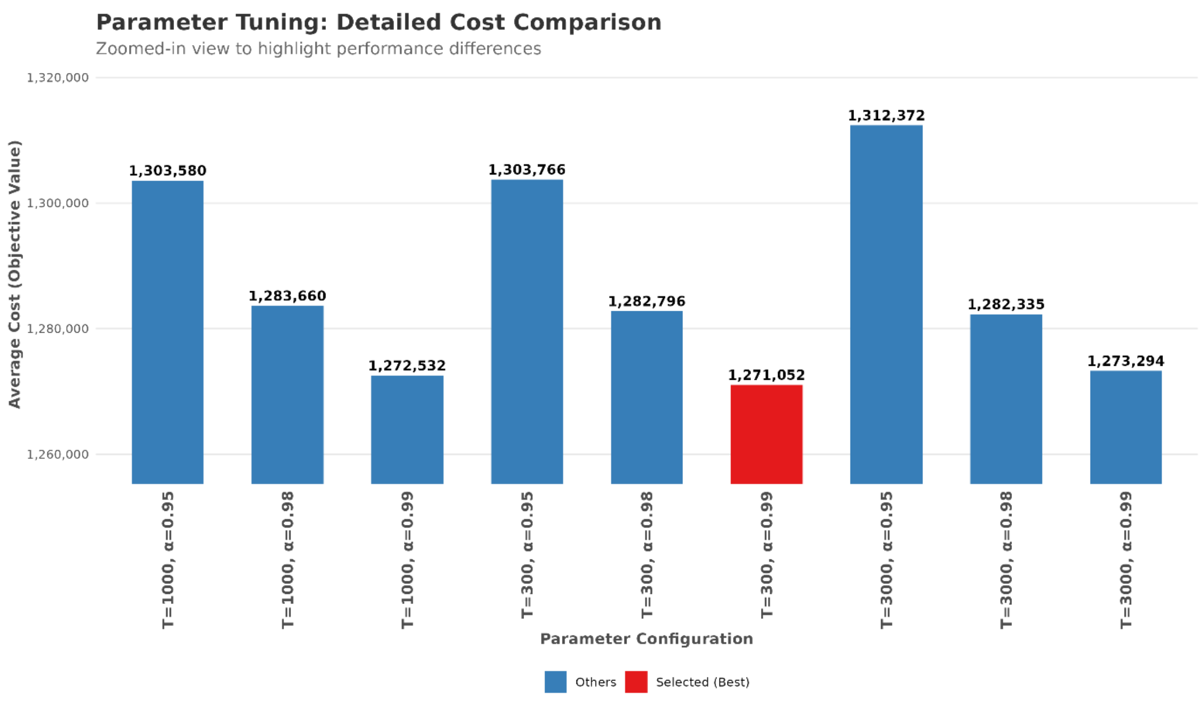


2. Interaction Effects (Interaction Plot)

The interaction plot highlights the relationship between the Initial Temperature and the Cooling Rate.

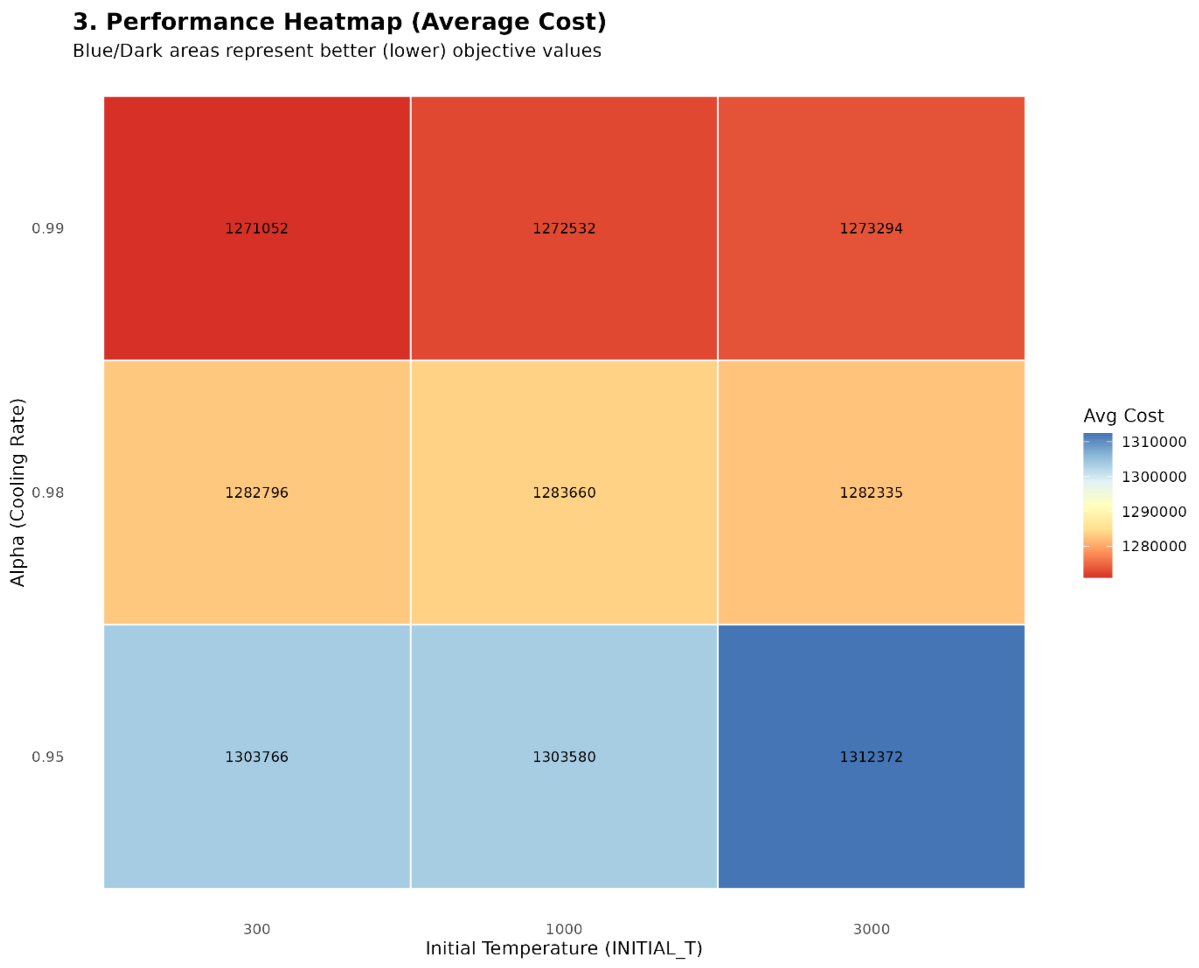
- **Robustness of High Alpha:** The blue line (α=0.99) is relatively flat, indicating that when the cooling rate is high, the algorithm is robust to changes in the initial temperature.
- **Sensitivity of Low Alpha:** The red line (α =0.95) shows a steep upward trend. This demonstrates that if a fast cooling schedule is chosen, increasing the initial temperature is detrimental to performance, likely because the algorithm "freezes" (converges) too quickly before effectively exploring the expanded search space.


Parameter Tuning: Average Cost Comparison
Performance of Different Temperature (T) and Cooling Rate (α) Combinations
Average Cost (Objective Value)
1,000,000
500,000
0
T=1000, α=0.95
T=1000, α=0.98
T=1000, α=0.99
T=300, α=0.95
T=300, α=0.98
T=300, α=0.99
T=3000, α=0.95
T=3000, α=0.98
T=3000, α=0.99
Parameter Configuration
Others
Selected (Best)


Parameter Tuning: Detailed Cost Comparison
Zoomed-in view to highlight performance differences
Average Cost (Objective Value)
1,320,000
1,300,000
1,280,000
1,260,000
1,303,580
1,283,660
1,272,532
1,303,766
1,282,796
1,271,052
1,312,372
1,282,335
1,273,294
T=1000, α=0.95
T=1000, α=0.98
T=1000, α=0.99
T=300, α=0.95
T=300, α=0.98
T=300, α=0.99
T=3000, α=0.95
T=3000, α=0.98
T=3000, α=0.99
Parameter Configuration
Others
Selected (Best)

3. Summary Heatmap

The performance heatmap consolidates these findings into a matrix view. The grid confirms that the optimal region for this problem lies in the upper section (representing α=0.99), where the lowest cost values (approx. 1.27M) are concentrated across all tested temperatures. In contrast, the bottom-right section (T=3000, α =0.95) represents the least desirable performance zone with costs exceeding 1.31M.

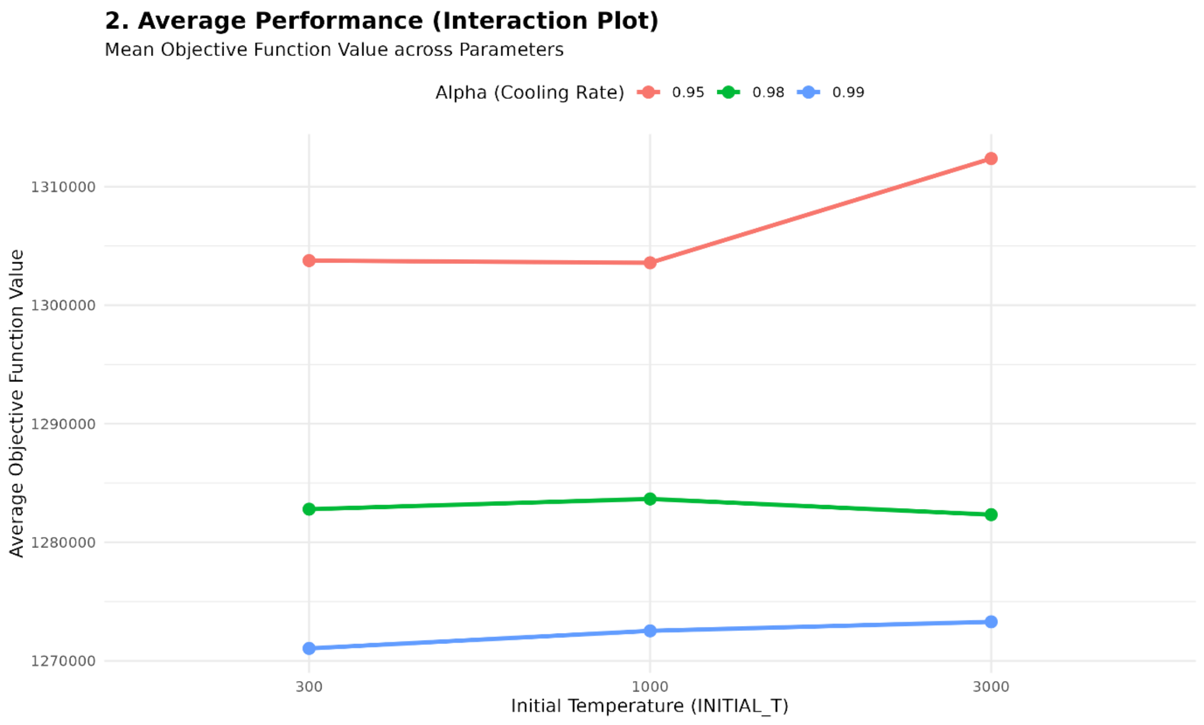


3. Parameter Tuning and Cost Comparison

A comprehensive grid search was conducted to identify the optimal hyperparameters for the algorithm, specifically testing combinations of Initial Temperature (T) and Cooling Rate (α).

- **Overall Performance:** The comparative analysis reveals that higher cooling rates generally result in lower average costs.

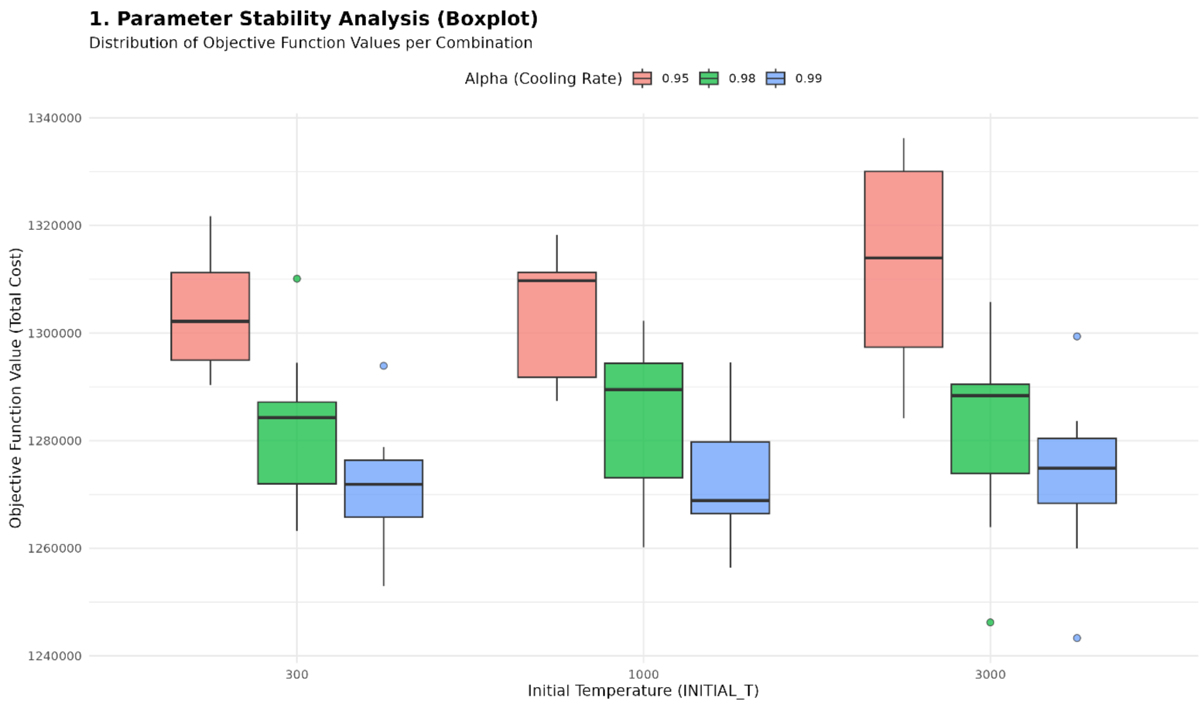
- **Optimal Configuration:** The detailed zoomed-in view identifies the configuration with **Initial Temperature (T) = 300** and **Cooling Rate α= 0.99** as the best performing model, achieving the lowest average cost of **1,271,052**.
- **Worst Configuration:** Conversely, configurations with a lower cooling rate (α=0.95), particularly combined with high initial temperatures (T=3000), resulted in the highest costs, exceeding 1,312,000.



4. Parameter Stability Analysis (Boxplot)

The boxplot analysis provides insight into the variance and reliability of each parameter combination.

- **Impact of Cooling Rate (α):** The plots clearly show that α = 0.99 (blue boxes) consistently produces not only lower median costs but also narrower interquartile ranges (tighter boxes) compared to α = 0.95 (pink boxes). This indicates that a slower cooling schedule leads to more stable and reproducible results.
- **Impact of Temperature (T):** At lower cooling rates (α = 0.95), increasing the initial temperature significantly increases the variability (wider whiskers), suggesting instability in the solution search.



4.5 Stress Test Code (Scalability and Robustness Analysis)

The third code performs a **stress test** to evaluate how the ALNS algorithm behaves as the problem size increases. Instead of fixing the number of universities, this test progressively increases the number of nodes and observes algorithm performance under heavier computational load.

The stress test runs the ALNS algorithm for different problem sizes:

- 50, 60, 70, 80, 90, and 100 nodes,
- with a fixed iteration limit of 2000 iterations for each case.

For each problem size, the code reports:

- the best cost found,
- total runtime,
- and feasibility status.

During execution, intermediate progress is logged at regular iteration intervals, showing steady improvement in solution quality and confirming convergence behavior. Across all tested sizes, the algorithm consistently produces **feasible solutions**, demonstrating strong robustness even as problem complexity increases.

The results show a clear and expected trend:

- as the number of nodes increases, runtime increases,
- solution costs grow proportionally with problem size,

- and convergence remains stable without feasibility violations.

This stress test confirms that the proposed ALNS framework scales well beyond the original problem size and remains reliable under increased computational demand.

5. Contribution to Sustainable Development Goals

The "College Tour Traveling Salesman Problem" project directly aligns with several United Nations Sustainable Development Goals (SDGs) by leveraging mathematical optimization to improve urban efficiency and reduce environmental impact.

SDG 9: Industry, Innovation, and Infrastructure

- Core Goal: Build resilient infrastructure, promote inclusive and sustainable industrialization, and foster innovation.
- Project Contribution: The project implements a sophisticated Adaptive Large Neighborhood Search (ALNS) metaheuristic. By modeling the non-Euclidean road map of Istanbul, this project provides an innovative logistical solution that can be applied to all large cities worldwide.

SDG 11: Sustainable Cities and Communities

- Core Goal: Make cities and human settlements inclusive, safe, resilient, and sustainable.
- Project Contribution: Efficient routing supports the development of sustainable transport systems. By optimizing a 20-day tour for 58 universities, the algorithm minimizes unnecessary vehicle presence in Istanbul, whci helps reduce the city’s severe traffic issue and improves the efficiency of all transportation methods

SDG 12: Responsible Consumption and Production

- Core Goal: Ensure sustainable consumption and production patterns.
- Project Contribution: The ALNS metaheuristic achieved a 45.64% improvement in route efficiency compared to random assignments. This optimization ensures that fuel and time are consumed responsibly, significantly reducing waste in the logistics lifecycle.

SDG 13: Climate Action

- Core Goal: Take urgent action to combat climate change and its impacts.
- Project Contribution: Transportation is a leading source of carbon emissions. By minimizing the total travel distance to 1,069.61 km, the algorithm directly reduces the carbon footprint associated with the tour, integrating climate-conscious planning into logistical operations

6. Conclusions

The objective of this project was to develop a feasible and efficient routing plan for a 20-day college tour visiting 58 universities in Istanbul, starting and ending each day at Taksim Square. By implementing the Adaptive Large Neighborhood Search (ALNS) metaheuristic, we successfully addressed the complexities of urban logistics, including time windows, daily travel limits, and university prioritization.

The experimental results demonstrate the effectiveness of the proposed approach:

Significant Optimization: The ALNS algorithm achieved a 45.64% improvement in total travel cost compared to the random initial baseline.

Robust Performance: Across 10 independent runs, the algorithm showed high consistency with a narrow 1.78% gap between the mean and the best-found solution.

Constraint Management: The model successfully managed strict university time windows and daily travel budgets. While minor overruns of the 8-hour daily limit occurred on three specific days, these were identified as strategic trade-offs where the penalty cost was lower than the cost of additional travel distance.

Solution Quality: Validation through warm-start experiments suggested that the best-found solution, covering a total distance of 1,069.61 km, resides in a high-quality local optimum that is difficult to improve upon further with standard operators.

In the end, this project shows that metaheuristic methods like ALNS can successfully solve large and complex routing problems in real life. The system created in this study can be used as a useful model for urban logistics, especially in big cities like Istanbul, where traffic, geography, and time limits make transportation more challenging.

7. References

[1] Lutz,R., (2014). "Adaptive Large Neighborhood Search." Bachelor’s thesis, Ulm University.

[2] Sarhani, M., & Voß, S. (2025). "Adaptive Neighborhood Search Based on Landscape Learning: A TSP Study." In Evolutionary Computation in Combinatorial Optimization (EvoCOP 2025), Lecture Notes in Computer Science, vol 15610. Springer, Cham.

[3] Sacramento, D., Pisinger, D., & Ropke, S. (2019). “An adaptive large neighborhood search metaheuristic for the vehicle routing problem with drones”. Transportation Research Part C:Emerging Technologies, 102, 289–315.

[4] Mara, S. T. W., Rifai, A. P., & Sopha, B. M. (2022). "An adaptive large neighborhood search heuristic for the flying sidekick traveling salesman problem with multiple drops." Expert Systems with Applications, 205, 117647.

[5] Keskin, M. (2014). An adaptive large neighborhood search approach for solving the electric vehicle routing problem with time windows (Master’s thesis). Sabancı University.

8. Appendix

This appendix lists the main source files and datasets used in the implementation and evaluation of the proposed Adaptive Large Neighborhood Search (ALNS) algorithm for the College Tour Traveling Salesperson Problem. The code and experimental artifacts are organized to separate algorithm execution, result analysis, and data storage.

1. README.md
Project description, problem definition, and instructions for running the experiments.

2. taksim-heuristics.ipynb
Main Jupyter Notebook containing the ALNS implementation for the College Tour TSP, including route construction, destroy–repair logic, and constraint handling.

3. graphs-heuristics.ipynb
Notebook used to generate convergence plots, comparison graphs, and visual analyses of the algorithm's performance.

4. outputs_alns_log_seed1002.csv
Iteration-level log file storing objective values and acceptance behavior during the ALNS runs.

5. tuning_results.csv
Summary of parameter tuning experiments, including average final cost, runtime, and feasibility statistics.

6. universite_distance_matrix_osrm_km_taksim.csv
Road-network-based distance matrix generated using OSRM, representing real driving distances between universities and the depot.

7. **stress_test / stress_test_outputs.txt**
Files used for stress testing and validating the robustness of the algorithm under different initial conditions.

The complete source code and experimental data are publicly available in the project's GitHub repository.