YEDITEPE UNIVERSITY
Faculty of Engineering
Department of Computer Engineering

Term Project Report

CSE480/591: Optimization with Metaheuristics

Student Name: Zeynep Sena Şeker

Student ID: 20210703007-072

Date: 31.10.2025

Project Title: Metaheuristic Approaches for the Capacitated Vehicle Routing Problem

Instructor: Asst. Prof. Dr. Gizem Süngü Terci
Fall 2025 Semester

1. Selected Problem and Motivation

We focus on *Capacitated Vehicle Routing Problem* (CVRP), where a depot dispatches a fleet of homogeneous vehicles with capacity Q to serve each customer exactly once and return to the depot. The goal is to minimize total travel cost (or distance) subject to capacity and routing feasibility constraints. CVRP is a canonical NP-hard problem with large practical impact in logistics and last-mile distribution, and it has sustained a rich body of exact and heuristic research over decades (Laporte, 2009; Toth and Vigo, 2002). Classic constructive baselines such as the Clarke–Wright savings method remain widely used and provide strong starting points for modern improvement heuristics (Clarke and Wright, 1964).

2. Formal Problem Definition

We study the Capacitated Vehicle Routing Problem (CVRP), in which a single depot dispatches a limited fleet of homogeneous vehicles with capacity Q to serve each customer exactly once and return to the depot. Moreover, each customer is served by exactly one vehicle in a single visit; the demand of a customer cannot be split across vehicles. Let $G = (V, A)$ be a complete directed graph with node set $V = \{0\} \cup N$, where 0 denotes the depot and $N = \{1, \dots, n\}$ the customers. Arc $(i, j) \in A$ bears a nonnegative travel cost (or distance) c_{ij} and, for each customer $i \in N$, demand $q_i > 0$. The number of available vehicles is p (upper bound). Following standard VRP modeling practice (Toth and Vigo, 2002; Laporte, 2009; Borčinová, 2017), we adopt a compact single-commodity flow (SCF) formulation. Additionally, we assume that the depot has sufficient inventory to satisfy $\sum_{i \in N} q_i$ and injects all commodities into the routes. Costs c_{ij} and demands q_i are deterministic and known; time windows, service times, driver hours, and other temporal constraints are not modeled unless stated otherwise.

2.1 Decision Variables

- Binary routing variables:

$$x_{rij} \in \{0, 1\} \quad \forall (i, j) \in A,\ i \neq j,$$

  where $x_{rij} = 1$ if and only if whether vehicle r ($r = 1, 2, \dots, p$) traverses arc (i, j).vehicle traverses arc (i, j).

2.2 Objective Function

The primary objective is to minimize total travel cost over all traversed arcs:

$$\min \quad Z = \sum_{r=1}^{p} \sum_{i=0}^{n} \sum_{\substack{j=0 \\ j \neq i}}^{n} c_{ij}\, x_{rij}. \tag{1}$$

2.3 Constraints

The model is subject to the following constraints.

(i) Customer service (degree constraints). Each customer j is visited exactly once by exactly one vehicle:

$$\sum_{r=1}^{p} \sum_{\substack{i=0 \\ i \neq j}}^{n} x_{rij} = 1, \qquad \forall j \in \{1, \ldots, n\}. \tag{2}$$

(ii) Depot departure. Each vehicle leaves the depot exactly once, starting its route:

$$\sum_{j=1}^{n} x_{r\,0j} = 1, \qquad \forall r \in \{1, \ldots, p\}. \tag{3}$$

(iii) Per-vehicle flow conservation For every node j and vehicle r, the number of arrivals equals the number of departures, ensuring route continuity (including return to depot):

$$\sum_{\substack{i=0 \\ i \neq j}}^{n} x_{rij} = \sum_{i=0}^{n} x_{rji}, \qquad \forall r \in \{1, \ldots, p\},\ \forall j \in \{0, \ldots, n\}. \tag{4}$$

(iv) Capacity The total demand served by each vehicle r cannot exceed vehicle capacity Q:

$$\sum_{i=0}^{n} \sum_{\substack{j=1 \\ j \neq i}}^{n} d_j\, x_{rij} \leq Q, \qquad \forall r \in \{1, \ldots, p\}. \tag{5}$$

(v) Subtour elimination. For any nonempty customer subset S, no closed tour can form entirely inside S without depot connection; it bounds the number of internal arcs:

$$\sum_{r=1}^{p} \sum_{i \in S} \sum_{\substack{j \in S \\ j \neq i}} x_{rij} \leq |S| - 1, \qquad \forall S \subseteq \{1, \ldots, n\},\ S \neq \emptyset. \tag{6}$$

(vi) Variable domains.

$$x_{rij} \in \{0,1\}, \qquad \forall r = 1,\dots,p,\ \forall i,j \in V,\ i \neq j. \tag{7}$$

3. Dataset or Benchmark Instances

We evaluate algorithms on publicly curated CVRP benchmarks from the CVRPLIB website and the X set of "New Benchmark Instances for the CVRP" (Uchoa et al., 2017). These resources provide standardized data files (coordinates, demands, capacity, best-known/optimal solutions) and consistent conventions for reporting results, facilitating reproducible comparisons across methods.

Core notation in the instances.

- $\boldsymbol{n}$: number of customers (excluding the depot). Instance names often embed n, e.g., `X-n284-k15` has $n = 284$ customers.
- $\boldsymbol{Q}$: per-vehicle capacity. Demands q_i satisfy $0 < q_i \leq Q$.
- $\boldsymbol{K}$: number of routes (vehicles) used by a solution. Some legacy sets fix K in advance (e.g., A/B/E/F/M/P series), while the `X` set does *not* fix K; instead it reports a lower bound $K_{\min} = \lceil \sum_{i \in N} q_i / Q \rceil$ (Uchoa et al., 2017).
- $\boldsymbol{K_{\min}}$: minimum possible number of routes from the capacity bound above. In fixed-K sets, the instance name uses `kB` to encode K (typically $K = K_{\min}$).
- **Tightness**: $\dfrac{\sum_{i \in N} q_i}{KQ}$ (only for fixed-K sets), indicating how close the total demand is to the available capacity (Uchoa et al., 2017).
- $\boldsymbol{r}$ and $\boldsymbol{n/K_{\min}}$: the `X` set controls *average route size* via a design parameter r so that $n/K_{\min}$ spans very short to very long routes, probing different algorithmic regimes (Uchoa et al., 2017).

Instance families and naming. Classical CVRPLIB families (A/B/E/F/M/P, CMT/Golden, etc.) cover Euclidean, clustered, and real-world layouts with n typically up to 200, sometimes > 400. The `X` family ($100 \leq n \leq 1000$) expands diversity by orthogonally varying depot location (central/eccentric/random), customer placement (random/clustered/RC), demand distributions (unitary, small/large with small/large variance, quadrant- dependent, SL), and average route size; coordinates lie on a $[0, 1000] \times [0, 1000]$ grid and Euclidean distances follow TSPLIB rounding (Uchoa et al., 2017). Instance names adopt the pattern `X-nA-kB`, with $A = n$ and $B = K_{\min}$.

Evaluation protocol. We follow the standard reporting practice used in CVRPLIB and (Uchoa et al., 2017):

1. **Cost:** total travel cost of the best solution found.
2. **#Routes (K):** routes used by the solution (compare with $K_{\min}$).
3. **GAP(%):** $100 \times (\text{Cost} - \text{BKS})/\text{BKS}$; if the optimal value is known, we use OPT instead of BKS.
4. **CPU time:** wall-clock time and basic hardware specs.
5. **Repetition:** for stochastic methods, average/best over multiple runs (e.g., 10–50), as in (Uchoa et al., 2017).

Subset used in this project. To balance difficulty and runtime, we select a stratified subset: small/medium cases ($n \in [20, 100]$), mid-scale ($n \in [100, 400]$), and large ($n \geq 400$), with varied depot/customer/demand attributes and a spread of $n/K_{\min}$. We keep Q and demand distributions as provided by the instances to preserve comparability. When fixed-K instances are used (e.g., `A/B/E/M` series), we report tightness alongside K.

4. Example Scenario/Problem Instance

In this section, we present a small but representative instance of the Capacitated Vehicle Routing Problem (CVRP). The instance consists of a single depot (node 0) and seven customers, each with a positive demand. All vehicles are homogeneous with a capacity of $Q = 50$. Distances are computed using Euclidean metrics, rounded to the nearest integer, following standard CVRPLIB conventions.

This compact instance retains the core structural characteristics of real CVRP cases: capacity restrictions, nontrivial routing decisions, and spatial dispersion of customers. Despite its small size, it allows meaningful evaluation of metaheuristic algorithms because it contains multiple feasible route combinations, route–capacity trade-offs, and non-obvious cost-minimizing configurations.

4.1 Instance Coordinates

Table 1 lists the coordinates of all nodes, including the depot. Coordinates are synthetic but consistent with the layout shown in Borčinová's (Borčinová, 2017) example.

Table 1: Node Coordinates

Node	**x**	**y**
0 (Depot)	50	50
1	30	70
2	70	75
3	85	60
4	80	40
5	60	30
6	40	25
7	20	45

4.2 Customer Demands

Customer demands are the same as those shown next to the nodes in Borčinová's (Borčinová, 2017) CVRP example (Figure 1 in the article).

Table 2: Customer Demands

Customer	**Demand** q_i
1	10
2	20
3	15
4	25
5	10
6	20
7	15

Vehicle capacity:

$$Q = 50.$$

4.3 Distance Matrix

Distances c_{ij} are computed as:

$$c_{ij} = \text{round}\left(\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}\right).$$

Table 3: Distance Matrix $C = (c_{ij})$

	0	1	2	3	4	5	6	7
0	0	28	32	36	32	22	27	30
1	28	0	40	56	58	50	46	27
2	32	40	0	21	36	46	58	58
3	36	56	21	0	21	39	57	67
4	32	58	36	21	0	22	43	62
5	22	50	46	39	22	0	21	43
6	27	46	58	57	43	21	0	28
7	30	27	58	67	62	43	28	0

4.4 Visual Representation of the Instance

Figure provides a TikZ illustration of the instance layout.

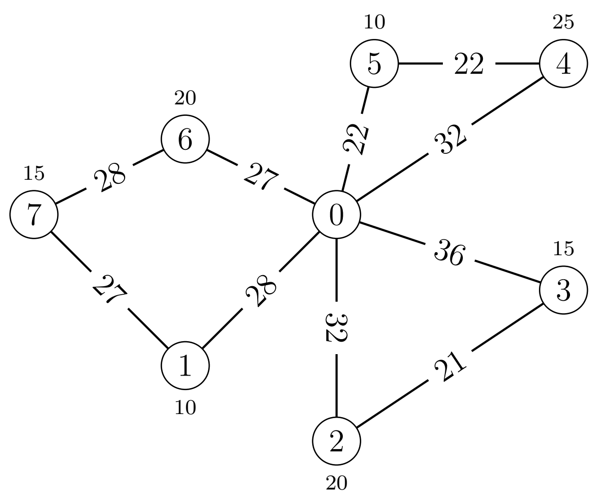

Figure 1: Feasible CVRP Example with $n = 7$ and $Q = 50$

For the given above Vehicle Routing Problem (VRP) instance, the optimal solution consists of three distinct routes originating from the depot. The first route, $0 \rightarrow 1 \rightarrow 7 \rightarrow 6 \rightarrow 0$, exactly utilizes the available capacity with a total demand of 45 units and yields a travel cost of 110. The second route, $0 \rightarrow 2 \rightarrow 3 \rightarrow 0$, services a combined demand of 35 units with a total cost of 89. The final route, $0 \rightarrow 5 \rightarrow 4 \rightarrow 0$, carries 35 units of demand and incurs a cost of 76. Summing these values, the total travel cost of the solution is 275. This result demonstrates that, under the given capacity constraint, the optimal routing structure naturally partitions the customer set into three balanced clusters, minimizing total distance while fully respecting vehicle capacity limitations.

5. Proposed Approach

The overall solution approach follows a two-stage structure consisting of customer clustering and intra-cluster route optimization. In the first stage, customers are partitioned into feasible vehicle routes using a sweep-based clustering method enhanced by Simulated Annealing. This metaheuristic allows the algorithm to escape local minima by probabilistically accepting non-improving moves during the early search phase. The annealing process starts from an initial sweep-based solution and iteratively refines cluster assignments as the temperature decreases.

In this study, two constructive–improvement hybrid algorithms are implemented for the Capacitated Vehicle Routing Problem (CVRP). This approach consists of two phases: an initial feasible solution construction and a subsequent local search improvement phase.

5.1 Overview Design and/or General Algorithm

The overall solution approach follows a two-stage structure consisting of customer clustering and intra-cluster route optimization. In the first stage, customers are partitioned into feasible vehicle routes using a sweep-based clustering method enhanced by Simulated Annealing. This metaheuristic allows the algorithm to escape local minima by probabilistically accepting non-improving moves during the early search phase. The annealing process starts from an initial sweep-based solution and iteratively refines cluster assignments as the temperature decreases.

After that, two hybrid heuristic approaches for solving the Capacitated Vehicle Routing Problem (CVRP). The proposed approach is based on the Clarke–Wright Savings heuristic, which constructs routes by iteratively merging initially independent customer routes according to potential distance savings, and subsequently applies a 2-opt local search to improve the internal structure of each route. The Savings-based approach exploits global cost reduction opportunities during construction. The integration of local search significantly enhances solution quality by eliminating inefficient routing patterns and converging to locally optimal solutions.

Algorithm 1 CVRP Solution Framework with Randomized Sweep and Simulated Annealing

for $run = 1$ **to** R **do**
    Step 1: Initial clustering using Randomized Sweep
    $clusters \leftarrow$ RANDOMIZED_SWEEP$(x, y, d, 0, C, D)$
    Step 2: Cluster improvement using Simulated Annealing
    $clusters \leftarrow$ SA_CLUSTERING$(clusters, d, C, D, T_0, T_{\min}, \alpha, ITERS_PER_T)$
    $total\_distance \leftarrow 0$
    $total\_iterations \leftarrow SA\_iterations$
    Step 3: Routing inside each cluster
    for each cluster c in $clusters$ **do**
        $tour \leftarrow$ CONSTRUCTION_HEURISTIC$(c, 0, D)$
        $tour \leftarrow$ IMPROVEMENT$(tour, D)$
        $total\_distance \leftarrow total\_distance +$ ROUTE_DISTANCE$(tour, D)$
        $total\_iterations \leftarrow total\_iterations + routing\_iterations$
    end for
    Store $total\_distance$, $total\_iterations$, and runtime
end for
Compute statistical measures over R runs
=0

5.2 Clustering Phase: Randomized Sweep with Simulated Annealing

Initially, customers are assigned to vehicle routes using a *randomized sweep* algorithm. Customer polar angles with respect to the depot are computed and randomly shifted to generate multiple sweep orders. Customers are sequentially assigned to clusters while respecting vehicle capacity constraints. This procedure is repeated multiple times, and the clustering with the minimum estimated travel cost is selected.

To further improve the clustering quality, *Simulated Annealing (SA)* is applied. In this phase, customers are probabilistically relocated between clusters while preserving feasibility. Moves that reduce total distance are always accepted, while worse moves may be accepted according to a temperature-dependent probability. This allows escaping local minima and leads to better-balanced and shorter routes. The clustering phase terminates when the temperature falls below a predefined threshold.

5.3 Algorithm: Clarke–Wright Savings Construction with 2-Opt Improvement

In this algorithm, the Clarke–Wright Savings heuristic is employed to initialize the solution by iteratively merging single-customer routes according to computed savings values.

This algorithm is single-solution-based and utilizes a local search neighborhood operator. The Savings-based approach employs a 2-opt operator that reverses subsequences within individual routes to reduce travel cost.

Candidate solutions generated by neighborhood moves are accepted if they produce a strictly lower total travel cost than the current solution and satisfy all capacity constraints. Non-improving or infeasible moves are rejected.

The solution is iteratively updated by applying improved neighborhood moves. After each accepted move, the solution structure and its objective value are updated, and the neighborhood is explored again based on the modified solution. The improvement process terminates when no further improving neighborhood move can be found, indicating convergence to a local optimum.

5.3.1 Clarke–Wright Savings Heuristic

The Clarke–Wright Savings algorithm constructs routes by iteratively merging initially independent routes. Initially, each customer is served by an individual route of the form:

$$(0 \rightarrow i \rightarrow 0), \quad \forall i \in N$$

The savings obtained by merging the routes of customers i and j is defined as:

$$s_{ij} = c_{i0} + c_{0j} - c_{ij}$$

All savings values are computed and sorted in descending order. Route merges are performed according to the highest savings values, subject to the following conditions:

- The total demand of the merged route does not exceed capacity Q.
- Customers i and j are located at the ends of their respective routes.

The algorithm terminates when no further feasible merges are possible.

5.3.2 2-Opt Route Improvement

After route construction, each route is independently improved using the 2-opt local search operator. The 2-opt method removes two edges and reconnects the route by reversing the order of the intermediate segment.

For a route

$$r = (0, i_1, \ldots, i_a, i_{a+1}, \ldots, i_b, i_{b+1}, \ldots, 0),$$

a 2-opt move replaces edges (i_a, i_{a+1}) and (i_b, i_{b+1}) with (i_a, i_b) and (i_{a+1}, i_{b+1}). The move is accepted if:

$$c_{i_a i_b} + c_{i_{a+1} i_{b+1}} < c_{i_a i_{a+1}} + c_{i_b i_{b+1}}$$

The process is repeated until no improving 2-opt move remains.

6. Data Structure and Fitness Function Implementation

6.1 Solution Representation

In this study, an individual solution to the Capacitated Vehicle Routing Problem is represented as a collection of vehicle routes. Each route is modeled as an ordered list of customer indices, starting and ending at the depot. Formally, a solution is defined as

$$S = \{r_1, r_2, \ldots, r_K\},$$

where each route

$$r_k = (0, i_1, i_2, \ldots, i_m, 0)$$

corresponds to the sequence of customers served by vehicle k, and K denotes the number of vehicles used. Each customer appears in exactly one route, and the order within each route defines the visit sequence.

The total demand served by a route r_k is constrained by the vehicle capacity:

$$\sum_{i \in r_k} q_i \leq Q.$$

This route-based representation is suitable for both constructive heuristics and local search operators, such as 2-exchange and 2-opt, which operate directly on customer sequences.

6.2 Fitness (Objective) Function

The fitness of a solution is evaluated by computing the total travel cost over all routes. For a given solution S, the objective function is defined as:

$$f(S) = \sum_{k=1}^{K} \sum_{(i,j) \in r_k} c_{ij},$$

where c_{ij} denotes the travel cost between nodes i and j. Lower fitness values correspond to better solutions.

Capacity constraints are enforced during solution construction and neighborhood exploration. Any candidate solution that violates the vehicle capacity constraint is considered infeasible and is either discarded or penalized, depending on the algorithm stage. This fitness evaluation framework ensures that all reported solutions are both cost-efficient and feasible with respect to problem constraints.

7. Experimental Study

This section describes the experimental environment and parameter settings used to evaluate the proposed heuristic algorithms for the Capacitated Vehicle Routing Problem (CVRP). All experiments were conducted under controlled and reproducible conditions.

7.1 Experimental Setup

All experiments were executed on a personal computer with the following specifications:

- **CPU:** Intel Core i5 (11th Generation)
- **RAM:** 16 GB
- **Operating System:** Windows 11 (64-bit)
- **Programming Language:** Python 3.x

The implementation relies on standard scientific computing libraries, including NumPy for numerical operations. No parallelization was employed.

7.1.1 Parameter Setting

Both algorithms are implemented as single-solution-based heuristics and share common stopping and evaluation mechanisms. The parameter settings used in the experiments are summarized below.

- **Vehicle Capacity (Q):** Change for all instance sets.
- **Simulated Annealing** The randomized sweep clustering algorithm was executed for 300 iterations to generate diverse initial clusterings. Simulated Annealing was applied with an initial temperature of $T_0 = 800$, a minimum temperature of $T_{\min} = 1$, a cooling rate of $\alpha = 0.995$, and 150 iterations per temperature level.
- **Initialization Method:**
  - Clarke–Wright Savings heuristic for the Savings + 2-opt algorithm
- **Local Search Operators:**
  - 2-opt for the Savings–based approach
- **Acceptance Criterion:** Strict improvement (only cost-reducing feasible moves are accepted)

- **Stopping Criteria:** No improving neighborhood move available

All parameter values were kept constant across runs to ensure fair comparison between the two algorithms.

7.1.2 Number of Runs

Each problem instance was independently solved **5 times** using each algorithm. For each instance, the best, mean, and standard deviation of the objective values, as well as computation times, were recorded and reported in the experimental results section.

7.2 Experimental Results

Table 4: Experimental Results Summary for CVRP Instances

Data	**Data Size**	**Capacity**	**Iterations**	**Time (s)**	**Total Distance**	**Optimal**
Rapor_DataSet	8	50	12	3.723	276.19	276.19
E-n23-k3	30	4500	601315	3.1856	574.02	569
E-n101-k14	100	112	3006896	5.8285	1200.01	1067
M-n200-k16	199	200	4692428	8.4094	1781.01	1274
Golden_20	420	200	10763585	11.5745	3293.92	1817.59
X-n856-k95	855	9	10213	0.0540	103764.86	88965
Loggi-n1001-k31	1000	180	8616547	43.9685	59238.29	284356

Table 5: Performance Statistics of Savings + 2-Opt

Data	**Best**	**Mean**	**Median**	**Std. Dev.**	**Med. Time (s)**	**Med. FEs**
Rapor_DataSet	276.19	276.19	276.19	0	10.0509	1380412
E-n23-k3	568.56	570.67	568.56	2.88	18.3508	1380880
E-n101-k14	1167.7	1183.85	1183.99	16.35	38.1454	1381346
M-n200-k16	1480.36	1511.65	1519.41	25.43	55.1887	1384224
Golden_20	2554.06	2672.74	2684.64	90.82	83.3751	1387265
X-n856-k95	105585.1	108686.07	109664.09	2262.41	192.3586	1393419
Loggi-n1001-k31	45634.86	46381.16	46116.12	644.14	190.6498	1449239

7.3 Final Routes Obtained by Savings + 2-Opt

7.3.1 E-n23-k3

- Route 1: $0 \to 7 \to 6 \to 5 \to 0$
- Route 2: $0 \to 3 \to 4 \to 0$
- Route 3: $0 \to 1 \to 2 \to 0$

7.3.2 E-n101-k14

- Cluster 1 tour: $0 \to 60 \to 5 \to 84 \to 17 \to 86 \to 16 \to 61 \to 6 \to 0$
- Cluster 2 tour: $0 \to 59 \to 91 \to 85 \to 93 \to 99 \to 96 \to 0$
- Cluster 3 tour: $0 \to 92 \to 37 \to 100 \to 38 \to 44 \to 98 \to 94 \to 0$
- Cluster 4 tour: $0 \to 13 \to 87 \to 42 \to 14 \to 97 \to 95 \to 0$
- Cluster 5 tour: $0 \to 53 \to 40 \to 73 \to 74 \to 22 \to 41 \to 15 \to 43 \to 57 \to 2 \to 58 \to 0$
- Cluster 6 tour: $0 \to 56 \to 23 \to 75 \to 72 \to 21 \to 0$
- Cluster 7 tour: $0 \to 4 \to 39 \to 67 \to 25 \to 55 \to 26 \to 0$
- Cluster 8 tour: $0 \to 54 \to 24 \to 29 \to 68 \to 80 \to 12 \to 28 \to 0$
- Cluster 9 tour: $0 \to 76 \to 77 \to 3 \to 79 \to 78 \to 34 \to 33 \to 0$
- Cluster 10 tour: $0 \to 50 \to 81 \to 9 \to 35 \to 65 \to 71 \to 51 \to 0$
- Cluster 11 tour: $0 \to 27 \to 69 \to 70 \to 30 \to 66 \to 20 \to 1 \to 0$
- Cluster 12 tour: $0 \to 31 \to 10 \to 32 \to 90 \to 63 \to 64 \to 62 \to 0$
- Cluster 13 tour: $0 \to 52 \to 7 \to 47 \to 49 \to 19 \to 11 \to 88 \to 0$
- Cluster 14 tour: $0 \to 18 \to 82 \to 48 \to 36 \to 46 \to 8 \to 45 \to 83 \to 0$
- Cluster 15 tour: $0 \to 89 \to 0$

7.3.3 M-n200-k16

- Route 1: 0 → 147 → 60 → 118 → 5 → 84 → 173 → 17 → 113 → 61 → 6 → 156 → 0
- Route 2: 0 → 183 → 85 → 141 → 86 → 16 → 93 → 99 → 104 → 96 → 0
- Route 3: 0 → 112 → 59 → 98 → 193 → 91 → 191 → 44 → 140 → 38 → 100 → 37 → 92 → 94 → 0
- Route 4: 0 → 95 → 97 → 151 → 192 → 119 → 14 → 43 → 142 → 42 → 172 → 144 → 87 → 117 → 13 → 0
- Route 5: 0 → 152 → 58 → 171 → 133 → 22 → 41 → 145 → 15 → 57 → 178 → 115 → 2 → 137 → 0
- Route 6: 0 → 53 → 40 → 73 → 74 → 75 → 23 → 186 → 56 → 197 → 72 → 21 → 180 → 0
- Route 7: 0 → 105 → 198 → 110 → 39 → 67 → 170 → 187 → 139 → 4 → 155 → 0
- Route 8: 0 → 26 → 149 → 195 → 179 → 54 → 130 → 165 → 55 → 25 → 24 → 134 → 177 → 109 → 12 → 0
- Route 9: 0 → 28 → 184 → 116 → 68 → 121 → 29 → 163 → 80 → 150 → 138 → 154 → 0
- Route 10: 0 → 76 → 196 → 77 → 3 → 158 → 79 → 129 → 169 → 78 → 34 → 164 → 185 → 0
- Route 11: 0 → 9 → 136 → 35 → 135 → 120 → 81 → 33 → 157 → 102 → 50 → 111 → 0
- Route 12: 0 → 176 → 51 → 103 → 161 → 71 → 65 → 66 → 188 → 20 → 122 → 1 → 0
- Route 13: 0 → 132 → 69 → 101 → 70 → 30 → 128 → 160 → 131 → 32 → 181 → 90 → 162 → 27 → 0
- Route 14: 0 → 127 → 190 → 159 → 126 → 63 → 108 → 189 → 10 → 31 → 167 → 0
- Route 15: 0 → 88 → 148 → 62 → 11 → 107 → 175 → 64 → 49 → 19 → 123 → 182 → 7 → 146 → 0
- Route 16: 0 → 52 → 153 → 106 → 194 → 82 → 48 → 168 → 143 → 36 → 47 → 124 → 46 → 8 → 18 → 0

- Route 17: 0 → 114 → 174 → 45 → 125 → 199 → 83 → 166 → 89 → 0

7.3.4 Golden_20

- Route 1: 0 → 60 → 5 → 84 → 17 → 86 → 16 → 61 → 6 → 0
- Route 2: 0 → 59 → 91 → 85 → 93 → 99 → 96 → 0
- Route 3: 0 → 92 → 37 → 100 → 38 → 44 → 98 → 94 → 0
- Route 4: 0 → 13 → 87 → 42 → 14 → 97 → 95 → 0
- Route 5: 0 → 53 → 40 → 73 → 74 → 22 → 41 → 15 → 43 → 57 → 2 → 58 → 0
- Route 6: 0 → 56 → 23 → 75 → 72 → 21 → 0
- Route 7: 0 → 4 → 39 → 67 → 25 → 55 → 26 → 0
- Route 8: 0 → 54 → 24 → 29 → 68 → 80 → 12 → 28 → 0
- Route 9: 0 → 76 → 77 → 3 → 79 → 78 → 34 → 33 → 0
- Route 10: 0 → 50 → 81 → 9 → 35 → 65 → 71 → 51 → 0
- Route 11: 0 → 27 → 69 → 70 → 30 → 66 → 20 → 1 → 0
- Route 12: 0 → 31 → 10 → 32 → 90 → 63 → 64 → 62 → 0
- Route 13: 0 → 52 → 7 → 47 → 49 → 19 → 11 → 88 → 0
- Route 14: 0 → 18 → 82 → 48 → 36 → 46 → 8 → 45 → 83 → 0
- Route 15: 0 → 89 → 0

7.3.5 Loggi-n1001-k31

- Route 1: 0 → 73 → 618 → 49 → 867 → 626 → 193 → 137 → 664 → 565 → 192 → 617 → 382 → 842 → 864 → 145 → 530 → 651 → 42 → 797 → 908 → 480 → 645 → 757 → 710 → 868 → 13 → 41 → 543 → 499 → 287 → 28 → 392 → 504 → 0
- Route 2: 0 → 955 → 31 → 109 → 597 → 104 → 217 → 295 → 378 → 939 → 337 → 231 → 782 → 923 → 735 → 952 → 819 → 956 → 771 → 712 → 718 → 72 → 942 → 247 → 624 → 865 → 346 → 111 → 353 → 116 → 131 → 336 → 80 → 692 → 0

- Route 3: 0 → 825 → 854 → 498 → 292 → 243 → 950 → 780 → 638 → 415 → 886 → 160 → 263 → 244 → 972 → 494 → 24 → 278 → 608 → 81 → 921 → 1000 → 578 → 709 → 912 → 765 → 275 → 899 → 583 → 531 → 88 → 0
- Route 4: 0 → 476 → 998 → 687 → 632 → 776 → 675 → 60 → 630 → 694 → 241 → 299 → 526 → 758 → 752 → 806 → 157 → 995 → 524 → 781 → 701 → 46 → 385 → 926 → 520 → 265 → 849 → 16 → 869 → 672 → 481 → 559 → 203 → 747 → 120 → 448 → 574 → 0
- Route 5: 0 → 773 → 807 → 459 → 245 → 491 → 442 → 697 → 826 → 430 → 590 → 820 → 99 → 567 → 786 → 472 → 571 → 273 → 977 → 601 → 529 → 728 → 206 → 599 → 232 → 75 → 633 → 59 → 290 → 703 → 270 → 612 → 86 → 0
- Route 6: 0 → 841 → 734 → 139 → 377 → 970 → 30 → 355 → 314 → 304 → 730 → 445 → 471 → 71 → 258 → 552 → 976 → 558 → 87 → 889 → 547 → 141 → 33 → 201 → 397 → 793 → 479 → 236 → 764 → 240 → 344 → 631 → 0
- Route 7: 0 → 726 → 615 → 902 → 341 → 367 → 174 → 823 → 828 → 853 → 410 → 951 → 582 → 64 → 234 → 395 → 1 → 418 → 676 → 528 → 660 → 824 → 982 → 396 → 331 → 924 → 682 → 105 → 138 → 598 → 822 → 259 → 980 → 373 → 306 → 399 → 518 → 0
- Route 8: 0 → 376 → 318 → 642 → 493 → 261 → 715 → 310 → 573 → 964 → 215 → 97 → 619 → 426 → 577 → 274 → 17 → 637 → 308 → 714 → 302 → 870 → 579 → 863 → 142 → 656 → 460 → 805 → 226 → 364 → 538 → 799 → 242 → 961 → 986 → 0
- Route 9: 0 → 156 → 678 → 434 → 305 → 409 → 251 → 118 → 8 → 856 → 455 → 741 → 451 → 477 → 536 → 219 → 439 → 114 → 739 → 595 → 783 → 50 → 293 → 38 → 818 → 223 → 755 → 127 → 199 → 394 → 502 → 312 → 775 → 829 → 228 → 0
- Route 10: 0 → 350 → 39 → 859 → 667 → 800 → 466 → 674 → 432 → 76 → 516 → 550 → 324 → 464 → 463 → 161 → 320 → 19 → 208 → 768 → 379 → 470 → 444 → 884 → 260 → 847 → 229 → 860 → 898 → 182 → 0
- Route 11: 0 → 602 → 144 → 569 → 501 → 252 → 339 → 424 → 500 → 129 → 391 → 319 → 607 → 35 → 511 → 686 → 668 → 6 → 485 → 422 → 542 → 566 → 512 → 393 → 766 → 997 → 358 → 269 → 221 → 271 → 365 → 662 → 0
- Route 12: 0 → 929 → 922 → 914 → 533 → 40 → 65 → 817 → 883 → 644 → 176 → 70 → 994 → 992 → 352 → 683 → 43 → 968 → 613 → 796 → 581 → 539 → 170 →

945 → 235 → 772 → 988 → 27 → 190 → 813 → 932 → 904 → 953 → 237 → 957 → 454 → 616 → 0

- Route 13: 0 → 549 → 309 → 130 → 411 → 532 → 276 → 748 → 284 → 47 → 52 → 61 → 568 → 808 → 561 → 218 → 343 → 497 → 985 → 457 → 905 → 327 → 713 → 366 → 636 → 316 → 721 → 482 → 606 → 711 → 0
- Route 14: 0 → 948 → 641 → 833 → 684 → 57 → 148 → 390 → 629 → 920 → 989 → 483 → 315 → 862 → 100 → 947 → 763 → 962 → 92 → 484 → 490 → 268 → 370 → 326 → 910 → 303 → 754 → 437 → 380 → 214 → 770 → 7 → 749 → 650 → 534 → 517 → 227 → 809 → 0
- Route 15: 0 → 429 → 213 → 172 → 934 → 254 → 496 → 699 → 509 → 408 → 433 → 406 → 106 → 720 → 169 → 839 → 647 → 858 → 769 → 855 → 388 → 848 → 723 → 564 → 570 → 167 → 112 → 963 → 26 → 67 → 238 → 698 → 0
- Route 16: 0 → 456 → 283 → 821 → 658 → 540 → 610 → 163 → 893 → 372 → 110 → 428 → 959 → 119 → 791 → 465 → 12 → 177 → 98 → 872 → 113 → 359 → 794 → 882 → 250 → 535 → 880 → 604 → 753 → 832 → 224 → 15 → 0
- Route 17: 0 → 917 → 21 → 716 → 401 → 777 → 334 → 178 → 576 → 885 → 450 → 386 → 233 → 151 → 103 → 915 → 330 → 978 → 66 → 184 → 188 → 255 → 876 → 486 → 348 → 404 → 335 → 414 → 620 → 220 → 523 → 333 → 966 → 600 → 960 → 62 → 84 → 834 → 0
- Route 18: 0 → 911 → 979 → 846 → 851 → 519 → 996 → 551 → 124 → 179 → 425 → 954 → 508 → 354 → 760 → 389 → 969 → 804 → 919 → 383 → 733 → 288 → 279 → 93 → 803 → 412 → 503 → 322 → 866 → 10 → 25 → 0
- Route 19: 0 → 453 → 740 → 593 → 790 → 361 → 132 → 3 → 194 → 594 → 918 → 171 → 150 → 744 → 115 → 446 → 374 → 659 → 548 → 22 → 51 → 423 → 506 → 928 → 688 → 881 → 931 → 407 → 546 → 938 → 168 → 155 → 784 → 369 → 473 → 152 → 96 → 840 → 0
- Route 20: 0 → 79 → 614 → 623 → 143 → 555 → 646 → 63 → 515 → 609 → 762 → 892 → 706 → 702 → 814 → 239 → 605 → 298 → 248 → 387 → 875 → 313 → 815 → 795 → 693 → 197 → 836 → 277 → 405 → 421 → 416 → 943 → 0
- Route 21: 0 → 20 → 557 → 468 → 165 → 603 → 787 → 212 → 731 → 652 → 54 → 661 → 981 → 475 → 665 → 357 → 222 → 246 → 91 → 356 → 82 → 983 → 525 → 495 → 685 → 452 → 857 → 537 → 622 → 166 → 347 → 742 → 0

- Route 22: 0 → 789 → 894 → 216 → 467 → 628 → 121 → 681 → 289 → 936 → 253 → 845 → 657 → 591 → 402 → 553 → 655 → 877 → 871 → 140 → 690 → 587 → 830 → 727 → 991 → 362 → 29 → 837 → 874 → 488 → 32 → 200 → 0
- Route 23: 0 → 625 → 584 → 779 → 774 → 785 → 653 → 122 → 478 → 185 → 45 → 48 → 256 → 183 → 993 → 670 → 154 → 196 → 689 → 513 → 621 → 78 → 888 → 272 → 792 → 162 → 901 → 750 → 507 → 572 → 323 → 736 → 0
- Route 24: 0 → 4 → 126 → 44 → 974 → 925 → 887 → 301 → 281 → 375 → 381 → 679 → 891 → 128 → 37 → 541 → 510 → 94 → 294 → 123 → 810 → 761 → 722 → 717 → 264 → 11 → 117 → 18 → 77 → 307 → 941 → 663 → 719 → 297 → 0
- Route 25: 0 → 724 → 639 → 588 → 207 → 838 → 230 → 384 → 90 → 417 → 125 → 798 → 338 → 83 → 913 → 984 → 987 → 14 → 136 → 707 → 321 → 852 → 788 → 585 → 999 → 811 → 909 → 286 → 209 → 554 → 262 → 696 → 0
- Route 26: 0 → 522 → 447 → 691 → 332 → 74 → 363 → 971 → 257 → 211 → 101 → 186 → 53 → 5 → 398 → 175 → 812 → 427 → 930 → 191 → 635 → 654 → 342 → 58 → 596 → 944 → 965 → 725 → 108 → 469 → 164 → 368 → 328 → 146 → 0
- Route 27: 0 → 431 → 589 → 36 → 311 → 648 → 189 → 560 → 737 → 527 → 187 → 879 → 149 → 102 → 700 → 249 → 266 → 413 → 627 → 56 → 9 → 756 → 55 → 198 → 705 → 973 → 937 → 843 → 767 → 487 → 291 → 23 → 267 → 801 → 458 → 677 → 0
- Route 28: 0 → 435 → 778 → 351 → 916 → 180 → 666 → 751 → 440 → 827 → 575 → 967 → 743 → 210 → 562 → 282 → 489 → 850 → 861 → 181 → 732 → 329 → 611 → 514 → 802 → 403 → 844 → 946 → 897 → 586 → 746 → 340 → 900 → 438 → 634 → 0
- Route 29: 0 → 643 → 592 → 202 → 300 → 669 → 708 → 563 → 296 → 443 → 441 → 147 → 449 → 95 → 325 → 903 → 990 → 34 → 492 → 158 → 204 → 420 → 107 → 649 → 2 → 159 → 704 → 133 → 729 → 958 → 225 → 0
- Route 30: 0 → 949 → 360 → 195 → 935 → 816 → 371 → 317 → 349 → 205 → 68 → 436 → 89 → 474 → 831 → 940 → 680 → 173 → 505 → 419 → 135 → 153 → 671 → 461 → 462 → 285 → 69 → 400 → 0
- Route 31: 0 → 521 → 759 → 835 → 895 → 927 → 280 → 878 → 933 → 890 → 134 → 345 → 873 → 544 → 745 → 896 → 673 → 738 → 907 → 556 → 85 → 640 → 975 → 695 → 545 → 906 → 580 → 0

7.3.6 Rapor_DataSet

- Route 1: 0 → 7 → 6 → 5 → 0
- Route 2: 0 → 3 → 4 → 0
- Route 3: 0 → 1 → 2 → 0

7.3.7 X-n856-k95

- Route 1: 0 → 62 → 90 → 162 → 82 → 313 → 786 → 50 → 525 → 558 → 798 → 547 → 568 → 512 → 575 → 565 → 799 → 611 → 482 → 678 → 804 → 775 → 753 → 780 → 470 → 485 → 854 → 425 → 577 → 157 → 745 → 582 → 832 → 101 → 672 → 741 → 473 → 551 → 440 → 310 → 618 → 595 → 498 → 490 → 713 → 640 → 465 → 573 → 696 → 570 → 746 → 355 → 507 → 159 → 327 → 66 → 255 → 175 → 73 → 300 → 403 → 555 → 77 → 602 → 563 → 168 → 537 → 172 → 38 → 331 → 401 → 153 → 394 → 107 → 83 → 109 → 194 → 106 → 85 → 174 → 271 → 43 → 628 → 201 → 2 → 642 → 64 → 349 → 253 → 414 → 59 → 22 → 105 → 225 → 831 → 164 → 0
- Route 2: 0 → 204 → 356 → 148 → 409 → 214 → 781 → 276 → 12 → 613 → 435 → 710 → 466 → 479 → 281 → 631 → 649 → 428 → 793 → 89 → 469 → 674 → 574 → 461 → 805 → 669 → 828 → 583 → 523 → 515 → 731 → 617 → 623 → 728 → 698 → 653 → 156 → 210 → 170 → 820 → 457 → 452 → 608 → 476 → 681 → 448 → 483 → 552 → 472 → 261 → 447 → 562 → 811 → 750 → 656 → 519 → 743 → 561 → 690 → 693 → 455 → 591 → 442 → 368 → 833 → 842 → 154 → 32 → 721 → 273 → 549 → 432 → 839 → 5 → 535 → 533 → 637 → 800 → 715 → 660 → 593 → 195 → 806 → 36 → 673 → 567 → 675 → 27 → 664 → 111 → 471 → 467 → 147 → 244 → 230 → 146 → 0
- Route 3: 0 → 209 → 264 → 68 → 236 → 353 → 320 → 138 → 272 → 381 → 17 → 376 → 125 → 93 → 42 → 173 → 296 → 364 → 449 → 411 → 372 → 422 → 286 → 108 → 400 → 208 → 150 → 133 → 335 → 226 → 311 → 220 → 399 → 78 → 39 → 309 → 53 → 47 → 419 → 212 → 302 → 415 → 290 → 140 → 316 → 383 → 41 → 122 → 377 → 165 → 9 → 200 → 202 → 240 → 113 → 262 → 25 → 334 → 410 → 134 → 49 → 20 → 52 → 263 → 413 → 427 → 792 → 179 → 250 → 351 → 393 → 216 → 666 → 453 → 348 → 747 → 496 → 665 → 44 → 706 → 500 → 638 → 283 → 181 → 247 → 412 → 416 → 732 → 251 → 136 → 384 → 171 → 37 → 324 → 275 → 197 → 0

- Route 4: 0 → 258 → 354 → 726 → 587 → 492 → 7 → 838 → 541 → 661 → 688 → 524 → 513 → 751 → 764 → 802 → 579 → 478 → 301 → 797 → 758 → 837 → 846 → 571 → 29 → 531 → 517 → 739 → 15 → 572 → 229 → 807 → 433 → 468 → 8 → 387 → 362 → 16 → 222 → 119 → 668 → 319 → 539 → 374 → 454 → 790 → 508 → 711 → 589 → 607 → 560 → 777 → 94 → 725 → 702 → 421 → 369 → 528 → 248 → 110 → 819 → 177 → 644 → 729 → 604 → 245 → 390 → 191 → 18 → 265 → 217 → 232 → 74 → 176 → 98 → 196 → 55 → 352 → 402 → 287 → 51 → 34 → 418 → 63 → 139 → 284 → 375 → 635 → 13 → 841 → 564 → 277 → 584 → 514 → 137 → 850 → 0

- Route 5: 0 → 765 → 557 → 526 → 630 → 709 → 759 → 580 → 770 → 459 → 481 → 769 → 620 → 795 → 823 → 776 → 520 → 650 → 444 → 813 → 345 → 338 → 757 → 480 → 767 → 824 → 783 → 488 → 773 → 744 → 737 → 151 → 629 → 566 → 677 → 646 → 648 → 624 → 659 → 437 → 487 → 830 → 585 → 590 → 522 → 720 → 10 → 544 → 503 → 182 → 760 → 851 → 439 → 662 → 521 → 124 → 462 → 816 → 361 → 545 → 596 → 491 → 844 → 639 → 446 → 680 → 667 → 695 → 554 → 318 → 458 → 676 → 712 → 486 → 495 → 733 → 707 → 460 → 836 → 809 → 434 → 592 → 429 → 632 → 670 → 705 → 848 → 671 → 817 → 505 → 654 → 794 → 763 → 752 → 779 → 443 → 0

- Route 6: 0 → 19 → 112 → 65 → 280 → 298 → 329 → 407 → 141 → 347 → 682 → 598 → 363 → 822 → 213 → 203 → 35 → 28 → 211 → 126 → 218 → 366 → 199 → 187 → 359 → 686 → 235 → 365 → 332 → 224 → 314 → 207 → 821 → 6 → 405 → 100 → 358 → 24 → 346 → 69 → 188 → 88 → 231 → 834 → 178 → 270 → 259 → 330 → 367 → 215 → 267 → 61 → 785 → 852 → 417 → 569 → 337 → 76 → 184 → 115 → 679 → 436 → 297 → 130 → 144 → 266 → 734 → 206 → 603 → 420 → 658 → 463 → 719 → 687 → 614 → 285 → 198 → 308 → 75 → 622 → 791 → 257 → 426 → 556 → 510 → 700 → 95 → 406 → 123 → 621 → 532 → 234 → 391 → 424 → 538 → 357 → 0

- Route 7: 0 → 4 → 56 → 379 → 708 → 431 → 748 → 326 → 14 → 306 → 724 → 307 → 581 → 323 → 155 → 21 → 610 → 619 → 827 → 529 → 249 → 382 → 789 → 685 → 509 → 787 → 849 → 826 → 475 → 716 → 771 → 167 → 736 → 643 → 812 → 740 → 506 → 397 → 576 → 651 → 504 → 540 → 548 → 663 → 718 → 445 → 714 → 704 → 697 → 536 → 530 → 782 → 772 → 474 → 749 → 683 → 727 → 818 → 456 → 305 → 499 → 559 → 723 → 129 → 612 → 441 → 527 → 121 → 336 → 128 → 166 → 396 → 219 → 553 → 79 → 342 → 274 → 325 → 546 → 315 → 118 → 339 → 494 → 278 → 143 → 26 → 717 → 120 → 252 → 142 → 641 → 228 → 398 → 192 → 292 → 241 → 0

- Route 8: 0 → 511 → 1 → 464 → 127 → 578 → 636 → 254 → 699 → 601 → 847 → 774 → 634 → 778 → 600 → 845 → 161 → 762 → 616 → 501 → 703 → 829 → 796 → 803 → 350 → 542 → 550 → 586 → 599 → 183 → 633 → 328 → 534 → 451 → 606 → 801 → 655 → 689 → 404 → 784 → 825 → 158 → 97 → 808 → 96 → 692 → 755 → 761 → 518 → 730 → 766 → 430 → 814 → 768 → 742 → 722 → 132 → 489 → 543 → 843 → 652 → 645 → 626 → 87 → 625 → 684 → 395 → 484 → 502 → 477 → 450 → 605 → 0
- Route 9: 0 → 855 → 0
- Route 10: 0 → 70 → 71 → 385 → 227 → 81 → 72 → 169 → 102 → 135 → 58 → 373 → 233 → 54 → 223 → 386 → 341 → 378 → 423 → 91 → 239 → 288 → 312 → 190 → 45 → 371 → 180 → 116 → 317 → 221 → 189 → 304 → 321 → 360 → 380 → 92 → 647 → 840 → 104 → 185 → 291 → 149 → 333 → 48 → 303 → 691 → 627 → 344 → 815 → 269 → 343 → 163 → 438 → 299 → 392 → 588 → 103 → 389 → 322 → 145 → 67 → 193 → 756 → 594 → 694 → 788 → 516 → 657 → 835 → 597 → 160 → 701 → 293 → 131 → 117 → 810 → 408 → 493 → 295 → 282 → 242 → 260 → 754 → 609 → 853 → 735 → 738 → 615 → 30 → 388 → 23 → 80 → 152 → 294 → 57 → 60 → 0
- Route 11: 0 → 205 → 243 → 256 → 340 → 86 → 3 → 11 → 186 → 497 → 33 → 99 → 289 → 46 → 268 → 246 → 238 → 237 → 31 → 84 → 114 → 40 → 279 → 370 → 0

8. Discussion and Conclusion

8.1 Discussion

The Savings-based algorithm constructs initial feasible solutions for the Capacitated Vehicle Routing Problem by exploiting global distance information through pairwise route-merging savings. This strategy enables the algorithm to form well-structured routes early in the optimization process while respecting vehicle capacity constraints, including explicit departure from and return to the depot.

The experimental results indicate that the algorithm produces competitive solutions across all tested datasets. For small instances such as *Rapor_Data_Set* and *E-n23-k3*, the best, mean, and median objective values are identical or nearly identical, with a standard deviation close to zero. This behavior indicates rapid convergence and a stable search process for low-dimensional problem instances.

For medium and large-scale instances, including *E-n101-k14* and *M-n200-k16*, the algorithm maintains robust performance across multiple runs. The close proximity of mean and median objective values to the best-observed solutions indicates reliable convergence toward high-quality regions of the solution space. The non-zero standard deviation observed in these cases is attributable to the increased combinatorial complexity and the existence of multiple local optima.

For large benchmark instances such as *Golden_20*, *X-n856-k95*, and *Loggi-n1001-k31*, an important modeling distinction must be noted when interpreting solution quality. The reported optimal or best-known solutions for these datasets do not explicitly start and terminate at the depot node. In contrast, all solutions generated by the proposed Savings-based algorithm enforce depot-based routes, ensuring that every vehicle both departs from and returns to the depot. As a result, direct numerical comparisons with the published optimal values for these instances inherently favor the reference solutions, since depot travel costs are not included in their objective values.

The application of 2-opt local search plays a crucial role in improving intra-route efficiency by eliminating route crossings and reducing redundant travel. Additionally, the incorporation of Simulated Annealing enhances global exploration by probabilistically accepting non-improving moves during early iterations. This mechanism allows the algorithm to escape local minima and contributes to improved average solution quality, particularly for larger instances.

From a computational perspective, the Savings-based algorithm exhibits predictable scaling behavior. Execution time and the number of function evaluations increase with problem size, as expected, but remain within reasonable bounds. The observed increase in computational effort is accompanied by measurable improvements in solution quality, indicating a favorable trade-off between runtime and optimization performance.

8.2 Conclusion

This study examined a Savings-based optimization framework for solving the Capacitated Vehicle Routing Problem, augmented with 2-opt local search and Simulated Annealing. The algorithm effectively combines greedy route construction with stochastic refinement to produce high-quality routing solutions.

The experimental evaluation demonstrates that the algorithm consistently achieves low best, mean, and median objective values across a variety of benchmark instances. The relatively small standard deviation observed in most cases indicates stable convergence behavior and robustness against stochastic variation. These characteristics are particularly evident in small and medium-sized datasets, where the algorithm rapidly converges to high-quality solutions.

For large-scale datasets such as *Golden_20*, *X-n856-k95*, and *Loggi-n1001-k31*, the higher objective values obtained by the algorithm must be interpreted in light of depot enforcement. Unlike the published optimal or best-known solutions, which omit explicit depot departure and return, the proposed method enforces depot-based routing for all vehicles. This modeling choice ensures strict feasibility and practical applicability, at the cost of increased total distance relative to reference values.

In conclusion, the Savings-based algorithm represents a reliable and effective approach for CVRP, offering a balanced trade-off between computational effort and solution quality. Its modular design and strong empirical performance make it a suitable foundation for further hybridization and methodological extensions.

9. References

References

Borčinová, Z. (2017). Two models of the capacitated vehicle routing problem. *Croatian Operational Research Review*, 8(2):463–469.

Clarke, G. and Wright, J. W. (1964). Scheduling of vehicles from a central depot to a number of delivery points. *Operations Research*, 12(4):568–581.

Laporte, G. (2009). Fifty years of vehicle routing. *Transportation Science*, 43(4):408–416.

Toth, P. and Vigo, D., editors (2002). *The Vehicle Routing Problem*. SIAM Monographs on Discrete Mathematics and Applications. SIAM, Philadelphia, PA.

Uchoa, E., Pecin, D., Pessoa, A., Poggi, M., Vidal, T., and Subramanian, A. (2017). New benchmark instances for the capacitated vehicle routing problem. *European Journal of Operational Research*, 257(3):845–858.

10. Apendix

Algorithm 2 Randomized Sweep + Simulated Annealing + Savings + 2-Opt for CVRP

Require: Coordinates (x, y), demands d, depot 0, vehicle capacity C, distance matrix D, T_0, $T_{\min}$, α, $ITERS\_PER\_T$

Ensure: Set of optimized vehicle routes

Step 1: Randomized Sweep Clustering
$best\_cost \leftarrow \infty$, $best\_clusters \leftarrow \emptyset$
for $k = 1$ **to** 300 **do**
  $shift \leftarrow$ random angle in $[-\pi, \pi]$
  for all customers $i \neq 0$ **do**
    $angle[i] \leftarrow 2\big(y[i] - y[0],\, x[i] - x[0]\big) + shift$
  end for
  Sort customers by $angle[i]$
  $clusters \leftarrow \emptyset$, $current \leftarrow [0]$, $load \leftarrow 0$
  for all customers i in sorted order **do**
    if $load + d[i] \leq C$ **then**
      Append i to $current$
      $load \leftarrow load + d[i]$
    else
      Append 0 to $current$
      Add $current$ to $clusters$
      $current \leftarrow [0, i]$, $load \leftarrow d[i]$
    end if
  end for
  Append 0 to $current$; Add $current$ to $clusters$
  $cost \leftarrow$ total distance of $clusters$
  if $cost < best\_cost$ **then**
    $best\_cost \leftarrow cost$, $best\_clusters \leftarrow clusters$
  end if
end for

Step 2: Simulated Annealing Clustering Refinement
$current \leftarrow best\_clusters$, $best \leftarrow best\_clusters$
$current\_cost \leftarrow cost(current)$, $best\_cost \leftarrow current\_cost$
$T \leftarrow T_0$
while $T > T_{\min}$ **do**

for $iter = 1$ **to** $ITERS\_PER\_T$ **do**
  Select two different clusters c_1 and c_2
  Select random customer $v \in c_1$ with $v \neq 0$
  if moving v to c_2 satisfies capacity constraint **then**
    $neighbor \leftarrow$ move v from c_1 to c_2
    $\Delta \leftarrow cost(neighbor) - current\_cost$
    if $\Delta < 0$ **or** rand() $< \exp(-\Delta/T)$ **then**
      $current \leftarrow neighbor$
      $current\_cost \leftarrow current\_cost + \Delta$
      if $current\_cost < best\_cost$ **then**
        $best \leftarrow current$, $best\_cost \leftarrow current\_cost$
      end if
    end if
  end if
end for
$T \leftarrow \alpha \cdot T$
end while
Remove empty clusters from *best*

Step 3: Routing with Savings + 2-Opt (per cluster)
$Routes \leftarrow \emptyset$
for all clusters c in *best* **do**
  $nodes \leftarrow c \setminus \{0\}$
{Savings construction}
  Create one route $[0, i, 0]$ for each $i \in nodes$
  Compute savings $s(i, j) = D(i, 0) + D(0, j) - D(i, j)$ for all pairs (i, j)
  Sort savings in descending order
  while more than one route exists **do**
    Select pair (i, j) with largest saving
    if routes containing i and j can be merged **then**
      Merge the two routes
    end if
  end while
  $tour \leftarrow$ merged route
{2-Opt improvement}
  $bestTour \leftarrow tour$, $bestDist \leftarrow cost(bestTour)$
  $improved \leftarrow$ **true**

while *improved* **do**
  improved $\leftarrow$ **false**
  for $i = 1$ **to** $|tour| - 2$ **do**
    for $j = i + 1$ **to** $|tour| - 1$ **do**
      $newTour \leftarrow$ reverse segment between positions i and j
      if $cost(newTour) < bestDist$ **then**
        $bestTour \leftarrow newTour$, $bestDist \leftarrow cost(newTour)$
        improved $\leftarrow$ **true**
      end if
    end for
  end for
end while
Add $bestTour$ to *Routes*
end for
return *Routes* =0