Solving the VRPTW Evolutionary Methods and Gurobi

Thomas Martinod[1]

[1] *Heuristics Course, EAFIT University, Medellín, Colombia*

November 13, 2024

Introduction

In this work, we present three different methods to solve the Vehicle Routing Problem with Time Windows (VRPTW) [1]. The first method is a metaheuristic approach, el segundo incorpora una búsqueda basada en vecindarios al primero, while the third is an exact algorithm. The methods are as follows:

- Genetic Algorithm (GA)
- GA + Variable Neighborhood Descent (VND)
- Gurobi Solver

The GA

A genetic algorithm (GA) involves initializing a population, selecting individuals based on fitness, combining parents through crossover, introducing random changes with mutation, and replacing the old population with a new generation. These steps repeat until a stopping criterion is met.

Parent Selection

Parent Selection is performed using the roulette wheel method, where the probability of selecting an individual is proportional to its normalized fitness. The selection probability $P(\mathbf{x}_i)$ for an individual $\mathbf{x}_i$ is defined as:

$$P(\mathbf{x}_i) = \frac{F(\mathbf{x}_i)}{\sum_{\mathbf{x}_j \in \mathcal{P}} F(\mathbf{x}_j)} \tag{1}$$

where:

- $F(\mathbf{x}_i)$ is the normalized fitness of individual $\mathbf{x}_i$.
- $\mathcal{P}$ represents the current population.

This method favors individuals with higher fitness, increasing their chances of passing genetic information to the next generation.

Crossover Operator

The crossover operator combines genetic material from two parent solutions to create offspring, facilitating exploration of the solution space. In this work, we use the **Order Crossover (OX)** technique, where two cut points are chosen in the parent chromosomes (a chromosome is all of the routes flattened in a single list).

The segment between these points is copied from one parent, while the remaining genes are filled in order from the other parent, excluding duplicates. This process generates diverse offspring that retain partial structure from each parent, helping to preserve and propagate advantageous traits.

Mutation Operator

The mutation operator introduces random changes to individuals to maintain genetic diversity and prevent premature convergence. Common mutation methods include swap mutation, where two genes exchange positions, and insertion mutation, where a gene is removed and reinserted at a different position. Mutation is applied with a certain probability, known as the mutation rate, to help explore new areas of the solution space.

As a brief annotation, after a mutation, as the nature of the problem forces, we have to check if the mutation is feasible. Otherwise, it gets discarded (this also happens in the crossover mutation).

Stopping Criterion

In general, a genetic algorithm stops when a specified stopping condition is met. In our code, we use two stopping criteria:

- The maximum number of generations.
- The maximum computation time.

Once either of these criteria is met, the algorithm returns the individual with the lowest fitness value, defined as:

$$f(\mathbf{x}) = \alpha D(\mathbf{x}) + \beta K(\mathbf{x}), \tag{2}$$

where $D(\mathbf{x})$ represents the distance, $K(\mathbf{x})$ represents the number of vehicles, and α and β are weighting factors.

Variable Neighborhood Descent (VND)

The VND algorithm aims to find the local optimal solution $s^* \in S$ that minimizes the objective function $f : S \to \mathbb{R}$ over the feasible solution space S. It consists on the steps:

- **Initialization:** Start with an initial solution $s \in S$ and set $k = 1$.
- **Main Loop:** While $k \leq k_{\text{max}}$:

  Perform a local search in neighborhood $\mathcal{N}_k(s)$ to find the best neighboring solution s'.
  If $f(s') < f(s)$, update $s \leftarrow s'$ and reset $k = 1$; otherwise, increment k by 1.
- **Stopping Condition:** Stop when the time limit is reached.

Neighborhoods Used

- **Swap Between Routes:** Swaps customers between two routes if feasible.
- **Or-Opt Within Route:** Moves small segments within a route to improve structure.
- **Relocate Between Routes:** Moves a customer from one route to another.
- **2-Opt Within Route:** Reverses a segment within a route to shorten it.
- **2-Opt Across Routes:** Exchanges segments between two routes.
- **Merge Routes:** Merges two routes into one if feasible.

Mathematical Formulation of the Problem

The VRPTW involves a fleet of K vehicles serving n customers. Each vehicle has a fixed capacity Q, and all routes begin and end at the depot v_0.

Each customer is represented by a vertex v_i, where $i \in \{1, 2, \ldots, n\}$. Customer v_i has a demand q_i and is located at coordinates (x_i, y_i). The delivery time window for v_i is defined by $[e_i, l_i]$, where e_i and l_i denote the earliest and latest times at which service can begin, respectively.

If a vehicle arrives at customer v_i before e_i, it must wait until the time window opens to start the service. Conversely, if a vehicle arrives after l_i, it will not be able to serve customer v_i. Each customer requires a service time s_i.

The depot, located at (x_0, y_0), has no demand and an unlimited time window. The travel time or cost between customers i and j is given by the euclidean distance $d_{i,j}$ [2].

Objective Function

The objective of the VRPTW (Vehicle Routing Problem with Time Windows) is twofold:

1. To minimize the number of vehicles K used.
2. To minimize the total distance D traveled by the vehicles.

This can be formulated as follows:

$$\min K \quad \text{and} \quad \min D = \sum_{i=0}^{n} \sum_{j=0}^{n} \sum_{k=1}^{K} d_{i,j} \cdot x_{i,j}^{k} \tag{3}$$

where D represents the total distance traveled by all vehicles, with $d_{i,j}$ being the Euclidean distance between customers i and j, and $x_{i,j}^{k}$ indicating whether vehicle k travels directly from i to j.

Constraints (Part 1)

The objective function is subject to the following constraints:

$$\sum_{i=0}^{n} x_{i,j}^{k} = y_{j}^{k}, \quad \forall k = 1, \ldots, K, \ \forall j = 1, \ldots, n \tag{4}$$

$$\sum_{i=0}^{n} x_{i,j}^{k} = y_{i}^{k}, \quad \forall k = 1, \ldots, K, \ \forall i = 1, \ldots, n \tag{5}$$

$$\sum_{k=1}^{K} y_{i}^{k} = 1, \quad \forall i = 1, \ldots, n \tag{6}$$

$$\sum_{i=0}^{n} y_{i}^{k} \cdot q_{i} \leq Q, \quad \forall k = 1, \ldots, K \tag{7}$$

Constraints (Part 2)

Additional constraints for the model:

$$\sum_{k=1}^{K} y_0^k = K \tag{8}$$

$$t_i + w_j + s_i + d_{i,j} = t_j, \quad \forall i, j = 1, \ldots, n, \ i \neq j \tag{9}$$

$$e_j \leq t_j \leq l_j, \quad \forall j = 1, \ldots, n \tag{10}$$

$$w_j = \max\{e_i - (t_i + s_i + d_{i,j}), 0\}, \quad \forall i = 1, \ldots, n \tag{11}$$

Decision Variables

The model uses the following decision variables:

$$x_{i,j}^k = \begin{cases} 1, & \text{if vehicle } k \text{ travels directly from } i \text{ to } j \\ 0, & \text{otherwise} \end{cases} \tag{12}$$

$$y_i^k = \begin{cases} 1, & \text{if customer } i \text{ is served by vehicle } k \\ 0, & \text{otherwise} \end{cases} \tag{13}$$

The constraints in (4)–(6) ensure that each customer is visited by exactly one vehicle, and that each vehicle follows a valid route. Constraint (7) ensures that each vehicle does not exceed its capacity Q. The depot constraint (8) requires all routes to start from the depot. The time window constraints (9)–(11) manage the scheduling, where t_i represents the arrival time at node i, w_j is the waiting time at customer j, and s_i is the service time at customer i.

Single-Objective Formulation in Gurobi

Gurobi allows two approaches for solving this problem: a multi-objective approach to minimize both K (number of vehicles) and D (total distance), or a single-objective approach. Since minimizing K is prioritized over D, we will use the single-objective approach.

The variable K is bounded as follows:

$$K_{\min} := \left\lceil \frac{1}{Q} \sum_{i=1}^{n} q_i \right\rceil \leq K \leq n := K_{\max} \tag{14}$$

where $K_{\min}$ represents the minimum number of vehicles required to satisfy the total demand, and $K_{\max}$ is the upper bound where each customer is served by a separate vehicle.

To solve the problem, we will iterate over possible values of K from $K_{\min}$ to $K_{\max}$.

Pseudocode

The following pseudocode outlines the iterative process:

- Initialize $K = K_{\min}$.
- While $K \in [K_{\min}, K_{\max}]$:
  - Define the problem constraints for the current K.
  - Solve the model using relaxation and branch-and-bound in the primal form.
  - If the model is feasible:
    - Record the minimum feasible K and stop the process.
  - If the model is not feasible:
    - Increment K by 1 and continue.

This pseudocode allows us to first find the optimal K (minimum number of vehicles) and, for a given K, to then minimize D (total distance traveled).

Results for Constructive Initial Solutions

Instance Data				Constructive				GA					GA + VND				
Instance	n	LB_K	LB_D	K	D	GAP_K	GAP_D	K	D	GAP_K	GAP_D	t[s]	K	D	GAP_K	GAP_D	t[s]
VRPTW1	25	3	133.286	3	257.922	0.000	1.606	3	257.922	0.000	0.935	3.692	3	191.814	0.000	0.439	3.675
VRPTW2	25	1	184.187	2	328.895	1.000	1.319	2	328.895	1.000	0.786	3.790	2	215.543	1.000	0.170	3.800
VRPTW3	25	2	399.722	8	634.907	3.000	1.294	8	634.907	3.000	0.588	3.911	8	618.330	3.000	0.547	3.852
VRPTW4	25	1	314.469	2	745.996	1.000	1.695	2	745.996	1.000	1.372	3.758	2	526.835	1.000	0.675	3.887
VRPTW5	25	3	162.712	5	567.906	0.667	2.183	5	567.906	0.667	2.490	3.851	4	462.156	0.333	1.840	3.839
VRPTW6	25	1	189.481	2	681.966	1.000	2.823	2	681.966	1.000	2.599	3.792	2	432.297	1.000	1.281	3.974
VRPTW7	50	5	252.397	5	493.598	0.000	1.523	5	493.598	0.000	0.956	3.819	5	363.247	0.000	0.439	3.999
VRPTW8	50	2	306.019	2	515.023	0.000	0.950	2	515.023	0.000	0.683	3.802	2	444.961	0.000	0.454	4.481
VRPTW9	50	4	623.861	13	1256.124	2.250	2.010	13	1256.124	2.250	1.013	3.854	12	1060.045	2.000	0.699	4.382
VRPTW10	50	1	512.907	3	1435.358	2.000	2.439	3	1435.358	2.000	1.798	3.825	3	887.000	2.000	0.729	6.297
VRPTW11	50	5	299.824	9	1060.250	0.800	2.358	9	1060.250	0.800	2.536	3.874	8	972.635	0.600	2.244	4.395
VRPTW12	50	1	252.250	3	1391.370	2.000	3.407	3	1391.370	2.000	4.516	3.731	3	900.471	2.000	2.570	5.718
VRPTW13	100	10	549.659	10	889.047	0.000	1.131	10	889.047	0.000	0.617	3.812	10	828.937	0.000	0.508	4.483
VRPTW14	100	3	529.193	3	779.902	0.000	0.584	3	779.902	0.000	0.474	3.743	3	591.557	0.000	0.118	7.636
VRPTW15	100	8	872.746	22	1962.402	1.750	2.490	22	1962.402	1.750	1.249	3.886	21	1665.316	1.625	0.908	10.260
VRPTW16	100	2	684.174	5	2208.197	1.500	2.927	5	2208.197	1.500	2.228	3.760	5	1273.612	1.500	0.862	37.381
VRPTW17	100	9	665.344	18	2185.075	1.000	2.874	18	2185.075	1.000	2.284	3.875	17	1745.175	0.889	1.623	10.143
VRPTW18	100	2	628.503	5	2938.973	1.500	4.211	5	2938.973	1.500	3.676	3.755	5	1395.080	1.500	1.220	37.521
					Mean:	**1.081**	**2.101**		Mean:	**1.081**	**1.711**			Mean:	**1.025**	**0.963**	

Figure: Results for GA and GA + VND with constructive initial solutions.

Results for Reactive GRASP Initial Solutions

Instance Data				GRASP				GA					GA + VND				
Instance	n	LB_K	LB_D	K	D	GAP_K	GAP_D	K	D	GAP_K	GAP_D	t[s]	K	D	GAP_K	GAP_D	t[s]
VRPTW1	25	3	133.286	4	259.744	0.333	1.624	4	259.744	0.333	0.949	3.838	3	191.814	0.000	0.439	3.827
VRPTW2	25	1	184.187	2	215.543	1.000	0.520	2	215.543	1.000	0.170	3.853	2	215.543	1.000	0.170	3.860
VRPTW3	25	2	399.722	9	715.537	3.500	1.585	9	715.537	3.500	0.790	3.896	9	627.133	3.500	0.569	3.892
VRPTW4	25	1	314.469	4	608.271	3.000	1.198	4	608.271	3.000	0.934	3.653	4	464.375	3.000	0.477	3.908
VRPTW5	25	3	162.712	5	561.447	0.667	2.147	5	561.447	0.667	2.451	3.637	5	476.955	0.667	1.931	3.885
VRPTW6	25	1	189.481	3	558.563	2.000	2.131	3	558.563	2.000	1.948	3.613	3	361.241	2.000	0.906	3.924
VRPTW7	50	5	252.397	9	680.576	0.800	2.478	9	680.576	0.800	1.696	3.848	5	363.247	0.000	0.439	5.030
VRPTW8	50	2	306.019	5	657.717	1.500	1.490	5	657.717	1.500	1.149	3.666	3	361.797	0.500	0.182	5.735
VRPTW9	50	4	623.861	13	1232.530	2.250	1.953	13	1232.530	2.250	0.976	3.785	13	1061.213	2.250	0.701	4.413
VRPTW10	50	1	512.907	4	1124.526	3.000	1.694	4	1124.526	3.000	1.192	3.784	4	844.065	3.000	0.646	5.656
VRPTW11	50	5	299.824	9	1095.061	0.800	2.468	9	1095.061	0.800	2.652	3.839	9	973.480	0.800	2.247	4.235
VRPTW12	50	1	252.250	6	1032.763	5.000	2.271	6	1032.763	5.000	3.094	3.824	5	686.312	4.000	1.721	4.774
VRPTW13	100	10	549.659	16	1678.450	0.600	3.022	16	1678.450	0.600	2.054	3.795	11	892.250	0.100	0.623	10.179
VRPTW14	100	3	529.193	7	1343.666	1.333	1.728	7	1343.666	1.333	1.539	3.766	3	591.557	0.000	0.118	38.643
VRPTW15	100	8	872.746	22	1973.364	1.750	2.510	22	1973.364	1.750	1.261	3.997	22	1690.641	1.750	0.937	8.441
VRPTW16	100	2	684.174	7	1833.461	2.500	2.261	7	1833.461	2.500	1.680	3.827	7	1216.986	2.500	0.779	24.244
VRPTW17	100	9	665.344	18	2106.481	1.000	2.735	18	2106.481	1.000	2.166	3.879	17	1725.395	0.889	1.593	10.494
VRPTW18	100	2	628.503	6	1725.319	2.000	2.059	6	1725.319	2.000	1.745	3.874	6	1341.622	2.000	1.135	21.259
					Mean:	**1.835**	**1.993**		Mean:	**1.835**	**1.580**			Mean:	**1.553**	**0.867**	

Figure: Results for GA and GA + VND with GRASP initial solutions.

Results for ACO Initial Solutions

Instance Data				ACO				GA					GA + VND				
Instance	n	LB_K	LB_D	K	D	GAP_K	GAP_D	K	D	GAP_K	GAP_D	t[s]	K	D	GAP_K	GAP_D	t[s]
VRPTW1	25	3	133.286	4	230.351	0.333	1.327	4	230.351	0.333	0.728	3.908	3	191.814	0.000	0.439	4.049
VRPTW2	25	1	184.187	2	257.412	1.000	0.815	2	257.412	1.000	0.398	3.808	2	215.543	1.000	0.170	3.839
VRPTW3	25	2	399.722	9	633.437	3.500	1.289	9	633.437	3.500	0.585	3.823	9	627.133	3.500	0.569	3.848
VRPTW4	25	1	314.469	4	531.360	3.000	0.920	4	531.360	3.000	0.690	3.854	4	494.824	3.000	0.574	3.961
VRPTW5	25	3	162.712	5	545.655	0.667	2.059	5	545.655	0.667	2.354	3.815	4	462.156	0.333	1.840	3.791
VRPTW6	25	1	189.481	4	446.500	3.000	1.503	4	446.500	3.000	1.356	3.762	3	361.241	2.000	0.906	3.924
VRPTW7	50	5	252.397	7	440.278	0.400	1.250	7	440.278	0.400	0.744	3.783	5	363.247	0.000	0.439	4.091
VRPTW8	50	2	306.019	4	421.801	1.000	0.597	4	421.801	1.000	0.378	3.717	3	361.797	0.500	0.182	4.905
VRPTW9	50	4	623.861	15	1164.915	2.750	1.791	15	1164.915	2.750	0.867	3.845	12	1051.997	2.000	0.686	4.716
VRPTW10	50	1	512.907	7	983.105	6.000	1.355	7	983.105	6.000	0.917	3.828	6	823.637	5.000	0.606	5.070
VRPTW11	50	5	299.824	11	1124.058	1.200	2.560	11	1124.058	1.200	2.749	3.921	10	985.691	1.000	2.288	4.185
VRPTW12	50	1	252.250	7	979.574	6.000	2.103	7	979.574	6.000	2.883	3.865	5	689.522	4.000	1.733	4.661
VRPTW13	100	10	549.659	13	1071.047	0.300	1.567	13	1071.047	0.300	0.949	3.937	10	828.937	0.000	0.508	7.101
VRPTW14	100	3	529.193	5	955.949	0.667	0.941	5	955.949	0.667	0.806	3.792	3	591.557	0.000	0.118	10.913
VRPTW15	100	8	872.746	27	1995.108	2.375	2.548	27	1995.108	2.375	1.286	4.190	23	1738.941	1.875	0.992	9.773
VRPTW16	100	2	684.174	10	1572.657	4.000	1.797	10	1572.657	4.000	1.299	3.976	9	1198.309	3.500	0.751	21.989
VRPTW17	100	9	665.344	23	2162.244	1.556	2.834	23	2162.244	1.556	2.250	4.005	18	1722.448	1.000	1.589	10.877
VRPTW18	100	2	628.503	11	1801.119	4.500	2.194	11	1801.119	4.500	1.866	3.795	10	1295.237	4.000	1.061	20.179
					Mean:	**2.347**	**1.636**		Mean:	**2.347**	**1.284**			Mean:	**1.817**	**0.858**	

Figure: Results for GA and GA + VND with ACO initial solutions.

Results Summary

Initial results from a lower-quality constructive method, referred to as the "humble" method, were also used to emphasize the high dependency on GA and neighborhood-based methods. The summary of the GAP values and their improvements over the constructive method are presented in the table below.

Method	**GA** (GAP_K, GAP_D)	**GA + VND** (GAP_K, GAP_D)
Constructive Method	(1.081, 1.711)	(1.025, 0.963)
GRASP Method	(1.835, 1.580)	(1.553, 0.867)
ACO Method	(2.347, 1.284)	(1.817, 0.858)
"Humble" Method	(6.644, 2.359)	(3.744, 1.402)

Table: GAP values (GAP_K, GAP_D) for different methods using GA and GA + VND.

Results Summary II

After applying the metaheuristic with VND, the best results prioritizing K over D were obtained with the initial solutions from the constructive method, showing a 5% improvement in GAP_K and a 54% improvement in GAP_D.

Also the table above shows that most of the advantages of the hybrid mehtod are provided by the VND, and that the GA is extremely sensitive to the initial solution (see "Humble" metod results).

Parameters for Comparison

The parameters to be varied are those used by the Genetic Algorithm (GA). Specifically, we have the weights in the fitness function (α and β), the number of generations (NG), the population size ($\mathcal{P}$), the crossover rate (CR), and the mutation probability (PM). We will start with the following default parameter values:

$$\alpha = 1,\ \beta = 10^4,\ NG = 2 \times 10^5,\ \mathcal{P} = 10^4,\ CR = 0.9,\ PM = 0.2 \quad (15)$$

We will vary each parameter individually, using the constructive method and GA + VND (the best method found) as benchmarks to evaluate the results.

Effect of Varying α and β

The table below shows the results for the GAP in terms of the number of vehicles (GAP_K) and distance (GAP_D) obtained by varying the weights α (weight for distance) and β (weight for the number of vehicles) simultaneously.

α	β	GAP_K	GAP_D
0.1	10^5	1.0248	0.9626
1	10^4	1.0248	0.9626
10	10^3	1.0251	0.9623
10^2	10^2	1.0251	0.9623
10^3	10	1.0260	0.9611

Table: Results for GAP in number of vehicles and distance by varying α and β.

Notice that the GAP is extremely insensitive to α and β, due mainly to the VND.

Effects of Mutation Probability and Population Size

- For mutation probabilities PM greater than 0.5, the GA experiences a large gap in vehicle count, with an average $GAP_K = 5$, while GAP_D remains relatively stable at around 1.5. High mutation rates may lead to excessive random changes, disrupting convergence.
- The population size $\mathcal{P}$ significantly impacts computational complexity without improving solution quality beyond a size of 500. Specifically, increasing $\mathcal{P}$ by a factor of l results in a proportional increase in execution time for the GA on a given instance.

High mutation probabilities can introduce instability in vehicle counts, while large population sizes primarily affect computation time rather than solution quality. Finding an optimal PM and keeping $\mathcal{P}$ manageable are essential for efficiency.

Effects of Crossover Rate and Number of Generations

- The crossover rate CR is the least sensitive parameter, possibly due to limited diversity in the initial population. Average GAP values remained nearly unchanged until CR was reduced to 0.3, indicating that crossover rate has minimal impact on solution quality.
- The number of generations NG also shows low sensitivity beyond 1000 generations. However, reducing NG to 10, 50, or 100 significantly worsens the GA's performance, with GAP_K values ranging from 2 to 3 across instances. This suggests that a minimum number of generations is necessary for effective convergence.

While crossover rate CR has limited influence, a minimum number of generations NG is crucial for quality solutions. Low values of NG lead to premature convergence, reducing solution accuracy.

Gurobi Results

The table below shows the optimal results obtained with Gurobi [3] compared to those obtained by our algorithm.

Instance	Optimal		Obtained		% Error	
VRPTW	Routes	Distance	Routes	Distance	Routes	Distance
1	3	191.814	3	191.814	0.00%	0.00%
2	2	215.543	2	215.543	0.00%	0.00%
3	8	618.330	8	618.330	0.00%	0.00%
4	2	523.657	2	526.835	0.00%	0.61%
5	4	462.156	4	462.156	0.00%	0.00%
6	2	432.297	2	432.297	0.00%	0.00%
7	5	363.247	5	363.247	0.00%	0.00%
8	2	444.961	2	444.961	0.00%	0.00%
9	11	1100.719	12	1060.045	9.09%	3.69%
10	2	953.285	3	900.471	50.00%	5.54%

Table: Comparison of optimal routes and distances vs. obtained results.

Conclusions

- In this work, we explored the use of a Genetic Algorithm (GA) combined with Variable Neighborhood Descent (VND) to solve the Vehicle Routing Problem with Time Windows (VRPTW). We analyzed the impact of various parameters on solution quality.
- The GA helps explore a larger portion of the search space; however, it is the VND that significantly improves solution quality by fine-tuning promising solutions.
- The initial population lacked diversity, which made parameters like α, β, and others relatively insensitive. This limited the potential impact of parameter adjustments.
- We achieved seven optimal solutions, as verified by comparisons with the Gurobi solver, demonstrating the effectiveness of the proposed approach.

References

[1] J. C. R. Agudelo, *Métodos evolutivos*, Curso: Heurística – CM0439, 2024.

[2] Q. Wu, X. Xia, H. Song, *et al.*, "A neighborhood comprehensive learning particle swarm optimization for the vehicle routing problem with time windows," *Swarm and Evolutionary Computation*, vol. 84, p. 101 425, Feb. 2024, ISSN: 2210-6502. DOI: 10.1016/j.swevo.2023.101425. [Online]. Available: http://dx.doi.org/10.1016/j.swevo.2023.101425.

[3] Gurobi Optimization, LLC, *Gurobi optimizer reference manual*, Version 10.0, 2023. [Online]. Available: https://www.gurobi.com.