# A Multi-Start Tabu Search with Set Partitioning for the Green VRP

Atef Dridi[1], Dalila Tayachi [2], Aziz Moukrim[3] and Lamjed Ben Said[1]

*Abstract*— **This paper tackles the Green Vehicle Routing Problem (GVRP), where vehicles with limited driving range must visit customers while recharging at Alternative Fuel Stations (AFSs). We propose a Multi-Start Tabu Search with Set Partitioning (MSTS-SP) approach structured in two phases. In the first phase, MSTS-SP uses a new constructive heuristic, Randomized Sectoring with Repair, to generate diverse initial solutions, which are then improved through multiple independent tabu search runs. The high-quality routes found during these runs are collected into a global pool. In the second phase, an exact set partitioning model is applied to this pool to select the best combination of routes. Computational experiments on 52 GVRP benchmark instances show that MSTS-SP matches 46 known best solutions (88%) and improves upon the best known solution for one large instance. These results demonstrate that MSTS-SP offers a competitive balance between solution quality and computational efficiency compared to state-of-the-art methods.**

*Index Terms*— **Green VRP, set partitioning, combinatorial optimization**

## I. INTRODUCTION

In response to growing environmental concerns and increasingly strict governmental regulations on fossil fuel-powered vehicles, the transportation sector has been shifting toward the use of Alternative Fuel Vehicles (AFVs). According to the International Energy Agency (IEA), transportation was responsible for 24% of global carbon dioxide (CO2) emissions in 2023, highlighting its significant impact on climate change. AFVs, which include those powered by electricity, hydrogen, or other non-conventional fuels, generally suffer from limited fuel autonomy, longer refueling times, and a limited refueling infrastructure. These operational challenges have motivated the development of tailored vehicle routing models. One such model is the Green Vehicle Routing Problem (GVRP) proposed by [3]. The GVRP focuses on minimizing the total distance traveled by a fleet of AFVs while accounting for fuel tank limitations, maximum tour duration, and refueling among the routes at Alternative Fuel Stations (AFSs). To tackle the problem, the authors developed a Mixed Integer Program (MIP) and proposed two heuristics: a Modified Clarke and Wright Savings (MCWS) algorithm and a density-based clustering method. They also introduced a 52-instance benchmark set, with results showing both heuristics to be effective.

[1]Atef Dridi and Lamjed Ben Said are with Higher Institute of Management of Tunis, University of Tunis, Tunisia

[2]Dalila Tayachi is with Higher School of Commerce (ESCT), University of Mannouba, Tunisia

[3]Aziz Moukrim is with Université de technologie de Compiègne, CNRS, Heudiasyc (Heuristics and Diagnosis of Complex Systems), CS 60 319 - 60 203 Compiègne Cedex

An improved MIP formulation for the GVRP of [3] was later introduced in [7], and solved using a branch-and-cut algorithm enhanced with a simulated annealing procedure featuring four neighborhood operators. Compared to the model of [3], their approach achieved tighter lower bounds and optimally solved 22 out of 40 GVRP small-sized instances with 20 customers.

Building on the model of [3], the study in [12] introduced the Electric Vehicle Routing Problem with Time Windows (EVRPTW), formulated as a mixed-integer program aimed at minimizing the total distance traveled by Electric vehicles (EVs). To tackle this problem, they developed a hybrid metaheuristic that integrates Tabu Search (TS) with Variable Neighborhood Search (VNS/TS). When applied to the benchmark instances originally proposed by [3], the VNS/TS consistently outperformed the heuristics of [3] across all tested cases.

The EVRP with a nonlinear charging function (EVRP-NL) was proposed in [9], and addressed using a MIP. To solve the problem efficiently, a hybrid metaheuristic was developed, combining a concentration heuristic with an iterated local search.

A variant of the EVRPTW where recharging stations offer both charging and battery-swapping options was presented in [16]. The solution method involves a two-phase heuristic: first, a local search with bounding techniques fixes the number of station visits; then, a genetic algorithm is applied to compute the delivery routing costs.

In [13], the authors introduced the EVRP with Flexible Time Windows (EVRPFTW), which allows service outside customers strict time bounds, with penalties for earliness and lateness. The objective is to minimize total costs, including travel, vehicle usage, and time window violations. They proposed a column generation-based solution, refined with integer and linear programming. Experiments on EVRPTW benchmark instances showed that flexibility can lead to shorter routes, fewer vehicles, or both.

The Multi-Mode Hybrid EVRP (MMHEVRP) was studied in [11], involving routing Hybrid Electric Vehicles capable of operating in different drive modes. The authors formulated the problem as a MIP minimizing travel costs across modes and proposed a matheuristic combining VNS with mathematical programming. The method was tested on adapted Hybrid Electric Vehicle – Traveling Salesman Problem (HEV-TSP) instances and showed competitive performance compared to other exact methods.

Apart from vehicle-based models using AFVs and EVs mentioned above, another branch of GVRP research targets emission reduction through more accurate modeling of fuel consumption and CO2 emissions. A key example is the Pollution Routing Problem (PRP) introduced by [2], which integrates an emissions-based objective function that balances fuel costs, driver wages, and CO2 penalties based on factors like vehicle load and speed. More recently, a Time-Dependent GVRP variant (TDGVRP) was proposed in [8], which considers traffic congestion by modeling travel speeds—and thus emissions—as time-dependent, aiming to schedule routes that reduce both fuel consumption and carbon emissions. These studies often employ exact methods such as branch-price-and-cut algorithms. Other contributions in this stream include works by [5] and [14].

Our literature review highlights a lack of efficient approaches capable of consistently solving large GVRP instances with AFVs. To address this gap, we propose a new solution method for the GVRP formulated by [3], which combines multi-start strategies, tabu search, and set partitioning. The main contributions of our work are: (i) we introduce the Multi-Start Tabu Search with Set Partitioning (MSTS-SP) algorithm; (ii) we design a new initial solution heuristic for the GVRP, the Randomized Sectoring with Repair Heuristic (RSR); and (iii) we achieve state-of-the-art results on 88% of the GVRP benchmark proposed by [3], including one new best known solution on large instances.

The remainder of this paper is organized as follows: Section II presents the formulation of the GVRP. Section III describes the components of the MSTS-SP approach. Section IV details the experimental results. Finally, Section V concludes the paper and outlines directions for future work.

## II. PROBLEM FORMULATION

The Green Vehicle Routing Problem (GVRP) is modeled on a complete weighted graph $G = (V, E)$, where the set of vertices $V = \{0, I, F\}$ includes the depot (denoted by 0), a set of customers $I = \{1, \ldots, n\}$, and a set of alternative fuel stations (AFSs) $F = \{n+1, \ldots, n+a\}$, with $a \geq 0$. The edge set $E = \{(i, j) \mid i, j \in V, i \neq j\}$ contains all possible arcs connecting distinct vertex pairs.

Each arc $(i, j) \in E$ is assigned a non-negative distance $d_{ij}$ and a travel time $t_{ij}$. Every node $i \in V$ has an associated service time $p_i > 0$. If $i \in I$, then $p_i$ represents the service duration at customer $i$. If $i \in F$, then $p_i$ is the refueling duration at AFS $i$.

A homogeneous fleet of alternative fuel vehicles (AFVs) is available, each with a fuel tank of capacity $Q$. Vehicles are assumed to move at a constant speed between any two vertices. Fuel consumption is proportional to the traveled distance; specifically, traversing arc $(i, j)$ depletes $r \cdot d_{ij}$ units from the tank, where $r$ is the fixed consumption rate.

The objective of the GVRP is to determine a set of vehicle routes minimizing the total traveled distance, subject to the following constraints:

- Every customer is served exactly once by one vehicle.
- All vehicle routes start and finish at the depot.
- The fuel level must be non-negative upon arriving at any node.
- Each tour must respect a maximum duration limit $T_{\text{max}}$.
- Refueling at any visited AFS restores the vehicle's fuel tank to its full capacity $Q$.
- AFSs (including the depot) can be visited multiple times, without restriction.
- Each vehicle may refuel as often as needed along its route.

For the detailed mathematical formulation of this problem, the reader is referred to the original work of [3].

## III. A MULTI START TABU SEARCH WITH SET PARTITIONING

As a resolution method for the GVRP, we propose a multi-start tabu search with a final set partitioning optimization phase (MSTS-SP). This approach combines the intensification power of tabu search [6] with the diversification of multi-start strategies and the global optimization capability of set partitioning. The general structure of our approach is presented in Algorithm 1.

Our algorithm proceeds in two main stages. First, we apply a series of $k$ independent tabu search runs. Each run begins from a solution generated using a newly designed heuristic for the GVRP, the Randomized Sectoring with Repair heuristic (RSR), described in detail in Section A. Then, a standard Tabu Search (TS) procedure is applied for $K_{max}$ iterations. During each TS run, every improving solution encountered contributes its forming routes to a global route pool $\Omega$.

In the second stage, we apply a Set Partitioning (SP) formulation over the accumulated route pool $\Omega$ to select a combination of routes that jointly serve all customers exactly once while minimizing the total travel cost. This SP phase allows the algorithm to recombine high-quality routes discovered across different TS runs, potentially yielding better overall solutions than any individual run.

The algorithm terminates after the SP solution is computed and returned.

Algorithm 1: MSTS-SP general structure

```
1:  Initialization;
    $k$: number of TS restarts;
    $K_{max}$: number of TS iterations;
2:  $\Omega \leftarrow \emptyset$
3:  $i \leftarrow 1$
4:  while $i \leq k$ do
5:      Generate an initial solution $S_i$ using the Randomized Sectoring with Repair heuristic (RSR)
6:      Starting from $S_i$, perform $K_{max}$ TS iterations and add the routes of every improving solution met to $\Omega$
7:      $i \leftarrow i + 1$
8:  end while
9:  $S \leftarrow SetPartitioning(\Omega)$
10: return S
```

### A. Randomized Sectoring with Repair heursitic

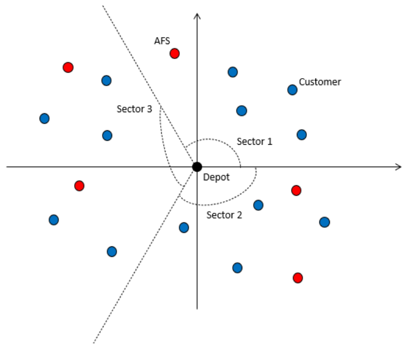
To generate an initial solution in each restart, we propose the Randomized Sectoring with Repair heuristic (RSR). This method is designed to generate diverse, good quality initial solutions through three key phases:

1) **Randomized Sector Partitioning**: The entire service space is partitioned into $m$ radial sectors originating from the depot (See Fig 1a). To introduce diversity between initial solutions, each restart rotates the sectors boundaries by a random angle $\theta \in [0°, 360°)$ (See Fig 1b).
2) **Biased-Randomized Route Construction**:
Within each sector, customers are not visited in strict order of proximity to the depot (as in classic greedy methods). Instead, starting from the depot, the next customer is selected randomly between the two nearest unvisited customers. This local randomization maintains reasonable route quality while injecting constructive randomness, which improves solution diversity across runs.
3) **Fuel Capacity Feasibility Repair:**
As the route is progressively built, the remaining fuel level is continuously monitored. If attempting to add a customer $c$ would result in the fuel level dropping below zero, a fuel capacity violation is detected. In this case, the algorithm identifies the critical path — the portion of the route between the last refueling station (or the depot, if no station has been visited yet) and the infeasible customer $c$.
Rather than applying a greedy repair by inserting the first feasible alternative fuel station (AFS) between the two customers where the violation occurs, we consider all insertion positions within the critical path. For each such position, we evaluate the insertion of the nearest available AFS and compute its impact on route feasibility and cost. Among these candidates:
   - Insertions that restore fuel feasibility are prioritized.
   - If multiple insertions restore feasibility, the one with the smallest increase in total distance is selected.

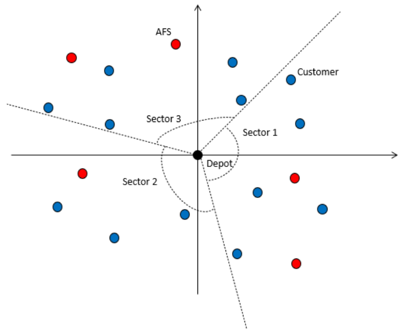
   If no insertion can restore feasibility within the segment, the current route is closed, and a new one is started from the depot. This repair mechanism emphasizes long-term feasibility over short-sighted gains, reducing the likelihood of early route termination and excessive route fragmentation.
   Additionally, two extra feasibility checks are performed:
   - After inserting any customer, we verify whether returning to the depot — either directly or via an AFS — is still feasible given the remaining fuel and time. Otherwise, the route is closed immediately.
   - If adding a customer causes the route to exceed the tour maximum duration, the route is also closed.

## *B. Tabu Search*

Each initial solution generated by the RSR heuristic is improved using a Tabu Search (TS) algorithm executed for a



(a) Service space partitioned into three $120°$ sectors originating from the depot



(b) Rotation by $\theta = 75°$

Fig. 1: Partitioning and rotating service space

Algorithm 2: Randomized Sectoring with Repair Heuristic

```
1:  Generate a random rotation angle $\theta \in [0°, 360°)$
2:  Partition the space into $m$ angular sectors around the depot, rotated by $\theta$
3:  **for** each sector $s$ **do**
4:      Initialize route $R \leftarrow$ [depot], fuel $\leftarrow Q$, time $\leftarrow 0$, unvisited $U \leftarrow$ customers in $s$
5:      **while** $U \neq \emptyset$ **do**
6:          Select customer $c$ at random among the two nearest unvisited customers to the last node in $R$
7:          **if** fuel $<$ distance to $c$ **then**
8:              Identify path between last AFS visit and $c$
9:              Insert best AFS within the path
10:             **if** no feasible insertion exists **then**
11:                 Close route $R$ and start a new one
12:                 **continue**            ▷ Skip to next iteration
13:             **end if**
14:         **end if**
15:         Append $c$ to $R$
16:         **if** (cannot return to depot) **or** (time $> T_{max}$) **then**
17:             Close route $R$ and start a new one
18:             **continue**
19:         **end if**
20:         Remove $c$ from $U$
21:     **end while**
22: **end for**
```

fixed number of iterations $K_{max}$. The TS procedure explores the solution space using five neighborhood structures: shift, swap, 2-opt, 2-opt* and swap AFS.

In each iteration, the algorithm generates the neighborhood of the current solution $S$, evaluates all feasible neighbor solutions, and selects the best non-tabu move to replace the current solution. If the selected move leads to an improved solution, the corresponding routes are added to a global route pool $\Omega$. The neighborhood operators used are defined as follows:

- **Shift:** Removes a node (customer or AFS) from its position in a route and inserts it in another route.
- **Swap:** Swaps two customers within or between routes.
- **2-opt:** Removes two non-adjacent edges within the same route and reconnects the route by reversing the order of the nodes between them.
- **2-opt$^*$:** Exchanges two arcs between different routes by reconnecting nodes across routes.
- **Swap AFS:** Swaps an AFS within a route with another.

### C. Set Partitioning

Once all $k$ TS runs are completed, the feasible GVRP routes collected in the global pool $\Omega$ are used to construct a final solution via a Set Partitioning Problem (SPP). This approach has been successfully adopted in various vehicle routing problems, including post-optimization strategies, such as in [17].

Let $x_r$ be a binary variable that indicates whether route $r \in \Omega$ is selected. Let $c_r$ be the distance traveled by route $r$, and $a_{ir}$ be equal to 1 if customer $i$ is served in route $r$, and 0 otherwise. The SPP is formulated as follows:

$$\begin{aligned} \min \quad & \sum_{r\in\Omega} c_r x_r \\ \text{s.t.} \quad & \sum_{r\in\Omega} a_{ir} x_r = 1 \quad \forall i \in I \\ & x_r \in \{0,1\} \quad \forall r \in \Omega \end{aligned}$$

Solving the SPP yields an optimal combination of routes that cover all customers exactly once while minimizing the total distance. The model is solved using the commercial CPLEX solver.

## IV. Numerical Results

All experiments were conducted on a laptop equipped with an Intel Core i5-7300HQ 2.50 GHz, 16 GB of RAM, and running a 64-bit version of Windows 10. The algorithm was implemented in Java and compiled using IntelliJ.

To evaluate the performance of our approach, we conducted experiments on both the small- and large-scale GVRP instances proposed in [3]. The small instances are organized into four sets, each containing ten 20-customer instances. These sets differ in the number of available charging stations (ranging from 2 to 10) and the spatial distribution of customers (clustered or random). The large instances are based on a real-world case study where the number of customers varies between 111 and 500 and the charging stations range between 21 and 28.

As baseline algorithms for comparison, we considered the following 4 algorithms: the Local Search algorithm (48A) of [4] (referring to the 48 combinations of the local search operators defined in [4]), the General Variable Neighborhood Search (GVNS/TS) of [10], the Improved Tabu Search (ITS) of [15] and the Multi-Start Local Search heuristic (MSLS) of [1].

It is worth noting that these algorithms were executed on generally faster machines than ours (Core i5, 2.50 GHz). Specifically, 48A ran on a Core i5 (2.80 GHz), GVNS/TS on a Core i7 (3.20 GHz), ITS on a Core i5 (2.50 GHz), and MSLS on a Core i5 (3.40 GHz). This provides helpful context for the CPU time comparisons that follow.

It is important to highlight that certain customers in the small and large instances of [3] cannot be served, either because they cannot be reached directly within the allowed maximum tour duration or would require multiple consecutive refueling stops to be served. In line with previous studies, we adopted the same preprocessing step by detecting and removing such customers from the instances.

Following a parameter tuning phase, the values of $k$ and $K_{max}$ were respectively fixed to 1000 and 100.

For each instance, the MSTS-SP was executed 10 times. We report the best solution obtained across the runs, the CPU time corresponding to that best solution, the average solution value over the 10 runs, and the number of vehicles $v$ used in the best solution. Tables I and II present the results for the small and large GVRP instances, respectively.

### A. Results on small instances

The results on the small instances show that our MSTS-SP algorithm delivers an excellent performance, matching all known best solutions (BKS) reported in the literature. As presented in Table I, MSTS-SP achieves an average gap of 0% from the BKS, outperforming the 48A approach (0.46% gap) and ITS (0.04%), and equaling the performance of GVNS/TS and MSLS. Notably, MSTS-SP is also considerably faster, with an average runtime of just 0.003 minutes—making it 2 to 5 times faster than the other methods.

Overall, these results confirm the efficiency and reliability of MSTS-SP for solving small-scale GVRP instances.

### B. Results on large instances

The results on large GVRP instances further demonstrate the effectiveness of our MSTS-SP algorithm. As shown in Table II, MSTS-SP matches the best known solutions (BKS) on 5 out of 12 large instances and improves the BKS for one of them (200c_21s), setting a new benchmark. It achieves an average gap of just 0.37%, outperforming ITS (5.41%), 48A (5.31%), and GVNS/TS (1.35%). While MSLS obtains a lower average gap of 0.15%, MSTS-SP remains competitive, achieving this performance with substantially lower computational effort.

In terms of CPU time, MSTS-SP solves all instances in an average of 26.47 minutes, which is significantly faster than

48A (157.03 min) and MSLS (49.71 min), while maintaining superior or comparable solution quality. Although GVNS/TS and ITS are slightly faster (13.22 and 13.09 min, respectively), they fail to reach comparable solution quality.

Overall, these results validate MSTS-SP as an efficient solver for large-scale GVRP instances.

## V. CONCLUSIONS

In this paper, we proposed a new multi-start tabu search with set partitioning (MSTS-SP) for the Green Vehicle Routing Problem (GVRP). A key feature of our method is a new constructive heuristic, called Randomized Sectoring with Repair, designed to efficiently generate diverse and feasible initial solutions. MSTS-SP builds a pool of high-quality routes using tabu search, then combines them through an exact set partitioning model. Computational experiments on GVRP instances demonstrate that MSTS-SP consistently produces high-quality solutions across both small and large instances, matching all known best solutions (BKS) on small instances and improving the BKS for one large instance. It also provides a good balance between solution quality and runtime compared to existing methods.

In future work, the approach can be extended to handle more complex GVRP variants, such as the GVRP with time windows or non-linear charging functions. Integrating machine learning to guide route selection during the search process could further enhance performance.

## REFERENCES

[1] Andelmin, Juho, and Enrico Bartolini. ”A multi-start local search heuristic for the green vehicle routing problem based on a multigraph reformulation.” Computers & Operations Research 109 (2019): 43-63.

[2] Bektaş, Tolga, and Gilbert Laporte. ”The pollution-routing problem.” Transportation Research Part B: Methodological 45.8 (2011): 1232-1250.

[3] Erdoğan, Sevgi, and Elise Miller-Hooks. ”A green vehicle routing problem.” Transportation research part E: logistics and transportation review 48.1 (2012): 100-114.

[4] Felipe, Ángel, et al. ”A heuristic approach for the green vehicle routing problem with multiple technologies and partial recharges.” Transportation Research Part E: Logistics and Transportation Review 71 (2014): 111-128.

[5] Franceschetti, Anna, et al. ”The time-dependent pollution-routing problem.” Transportation Research Part B: Methodological 56 (2013): 265-293.

[6] Glover, Fred. ”Tabu search—part I.” ORSA Journal on computing 1.3 (1989): 190-206.

[7] Koç, Çağrı, and Ismail Karaoglan. ”The green vehicle routing problem: A heuristic based exact solution approach.” Applied Soft Computing 39 (2016): 154-164.

[8] Luo, Hongyuan, Mahjoub Dridi, and Olivier Grunder. ”A branch-price-and-cut algorithm for a time-dependent green vehicle routing problem with the consideration of traffic congestion.” Computers & Industrial Engineering 177 (2023): 109093.

[9] Montoya, Alejandro, et al. ”The electric vehicle routing problem with nonlinear charging function.” Transportation Research Part B: Methodological 103 (2017): 87-110.

[10] Sadati, Mir Ehsan Hesam, and Bülent Çatay. ”A hybrid variable neighborhood search approach for the multi-depot green vehicle routing problem.” Transportation Research Part E: Logistics and Transportation Review 149 (2021): 102293.

[11] Seyfi, Majid, et al. ”Multi-mode hybrid electric vehicle routing problem.” Transportation Research Part E: Logistics and Transportation Review 166 (2022): 102882.

[12] Schneider, Michael, Andreas Stenger, and Dominik Goeke. ”The electric vehicle-routing problem with time windows and recharging stations.” Transportation science 48.4 (2014): 500-520.

[13] Taş, Duygu. ”Electric vehicle routing with flexible time windows: a column generation solution approach.” Transportation Letters 13.2 (2021): 97-103.

[14] Tayachi, D., and H. Boukadi. ”A variable neighborhood search to reduce carbon dioxide emissions in the capacitated vehicle routing problem.” 2019 6th international conference on control, decision and information technologies (CoDIT). IEEE, 2019.

[15] Tayachi, Dalila, and Atef Dridi. ”An Improved Tabu Search Algorithm for the Green Vehicle Routing Problem.” 2023 9th International Conference on Control, Decision and Information Technologies (CoDIT). IEEE, 2023.

[16] Verma, Amit. ”Electric vehicle routing problem with time windows, recharging stations and battery swapping stations.” EURO Journal on Transportation and Logistics 7.4 (2018): 415-451.

[17] Yahiaoui, Ala-eddine, Aziz Moukrim, and Mehdi Serairi. ”GRASP-ILS and set cover hybrid heuristic for the synchronized team orienteering problem with time windows.” International Transactions in Operational Research 30.2 (2023): 946-969.

| instance | BKS | 48A | | GVNS/TS | | | ITS | | | MSLS | | | MSTS-SP | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | $v$ | best | $v$ | best | Avg | $v$ | best | Avg | $v$ | best | Avg | $v$ | best | Avg |
| 20c3sU1 | 1797.49 | 6 | 1805.41 | 6 | **1797.49** | 1797.49 | 6 | **1797.49** | 1797.49 | 6 | **1797.49** | 1797.49 | 6 | **1797.49** | 1797.49 |
| 20c3sU2 | 1574.78 | 6 | **1574.78** | 6 | **1574.78** | 1574.78 | 6 | **1574.78** | 1574.78 | 6 | **1574.78** | 1574.78 | 6 | **1574.78** | 1574.78 |
| 20c3sU3 | 1704.48 | 6 | **1704.48** | 6 | **1704.48** | 1704.48 | 6 | **1704.48** | 1704.48 | 6 | **1704.48** | 1704.48 | 6 | **1704.48** | 1704.48 |
| 20c3sU4 | 1482 | 5 | **1482** | 5 | **1482** | 1482 | 5 | **1482** | 1482 | 5 | **1482** | 1482 | 5 | **1482** | 1482 |
| 20c3sU5 | 1689.37 | 6 | **1689.37** | 6 | **1689.37** | 1689.37 | 6 | **1689.37** | 1689.37 | 6 | **1689.37** | 1689.37 | 6 | **1689.37** | 1689.37 |
| 20c3sU6 | 1618.65 | 6 | **1618.65** | 6 | **1618.65** | 1618.65 | 6 | **1618.65** | 1618.65 | 6 | **1618.65** | 1618.65 | 6 | **1618.65** | 1618.65 |
| 20c3sU7 | 1713.66 | 6 | **1713.66** | 6 | **1713.66** | 1713.87 | 6 | **1713.66** | 1713.66 | 6 | **1713.66** | 1713.66 | 6 | **1713.66** | 1713.66 |
| 20c3sU8 | 1706.5 | 6 | 1722.78 | 6 | **1706.5** | 1706.5 | 6 | **1706.5** | 1706.5 | 6 | **1706.5** | 1706.5 | 6 | **1706.5** | 1706.5 |
| 20c3sU9 | 1708.82 | 6 | **1708.82** | 6 | **1708.82** | 1709.65 | 6 | **1708.82** | 1708.82 | 6 | **1708.82** | 1708.82 | 6 | **1708.82** | 1708.82 |
| 20c3sU10 | 1181.31 | 4 | **1181.31** | 4 | **1181.31** | 1181.31 | 4 | **1181.31** | 1181.31 | 4 | **1181.31** | 1181.31 | 4 | **1181.31** | 1181.31 |
| 20c3sC1 | 1173.57 | 4 | 1178.97 | 4 | **1173.57** | 1173.57 | 4 | **1173.57** | 1173.57 | 4 | **1173.57** | 1173.57 | 4 | **1173.57** | 1173.57 |
| 20c3sC2 | 1539.97 | 5 | **1539.97** | 5 | **1539.97** | 1539.97 | 5 | **1539.97** | 1539.97 | 5 | **1539.97** | 1539.97 | 5 | **1539.97** | 1539.97 |
| 20c3sC3 | 880.2 | 3 | **880.2** | 3 | **880.2** | 880.2 | 3 | **880.2** | 880.2 | 3 | **880.2** | 880.2 | 3 | **880.2** | 880.2 |
| 20c3sC4 | 1059.35 | 4 | **1059.35** | 4 | **1059.35** | 1059.94 | 4 | **1059.35** | 1059.35 | 4 | **1059.35** | 1059.35 | 4 | **1059.35** | 1059.35 |
| 20c3sC5 | 2156.01 | 7 | **2156.01** | 7 | **2156.01** | 2156.04 | 7 | **2156.01** | 2156.01 | 7 | **2156.01** | 2156.01 | 7 | **2156.01** | 2156.01 |
| 20c3sC6 | 2758.17 | 8 | **2758.17** | 8 | **2758.17** | 2758.17 | 8 | **2758.17** | 2758.17 | 8 | **2758.17** | 2758.17 | 8 | **2758.17** | 2758.17 |
| 20c3sC7 | 1393.99 | 4 | **1393.99** | 4 | **1393.99** | 1393.99 | 4 | **1393.99** | 1393.99 | 4 | **1393.99** | 1393.99 | 4 | **1393.99** | 1393.99 |
| 20c3sC8 | 3139.72 | 9 | **3139.72** | 9 | **3139.72** | 3139.72 | 9 | **3139.72** | 3139.72 | 9 | **3139.72** | 3139.72 | 9 | **3139.72** | 3139.72 |
| 20c3sC9 | 1799.94 | 6 | **1799.94** | 6 | **1799.94** | 1799.94 | 6 | **1799.94** | 1799.94 | 6 | **1799.94** | 1799.94 | 6 | **1799.94** | 1799.94 |
| 20c3sC10 | 2583.42 | 8 | **2583.42** | 8 | **2583.42** | 2583.42 | 8 | **2583.42** | 2583.42 | 8 | **2583.42** | 2583.42 | 8 | **2583.42** | 2583.42 |
| S1_2i6s | 1578.12 | 6 | **1578.12** | 6 | **1578.12** | 1578.12 | 6 | **1578.12** | 1578.12 | 6 | **1578.12** | 1578.12 | 6 | **1578.12** | 1578.12 |
| S1_4i6s | 1397.27 | 5 | 1413.97 | 5 | **1397.27** | 1397.27 | 5 | **1397.27** | 1397.27 | 5 | **1397.27** | 1397.27 | 5 | **1397.27** | 1397.27 |
| S1_6i6s | 1560.49 | 6 | 1571.3 | 5 | **1560.49** | 1560.49 | 5 | **1560.49** | 1560.49 | 5 | **1560.49** | 1560.49 | 5 | **1560.49** | 1560.49 |
| S1_8i6s | 1692.32 | 6 | **1692.32** | 6 | **1692.32** | 1692.32 | 6 | **1692.32** | 1692.32 | 6 | **1692.32** | 1692.32 | 6 | **1692.32** | 1692.32 |
| S1_10i6s | 1173.48 | 4 | **1173.48** | 4 | **1173.48** | 1173.48 | 4 | **1173.48** | 1173.48 | 4 | **1173.48** | 1173.48 | 4 | **1173.48** | 1173.48 |
| S2_2i6s | 1633.1 | 6 | 1645.8 | 6 | **1633.1** | 1633.1 | 6 | 1645.8 | 1645.8 | 6 | **1633.1** | 1633.1 | 6 | **1633.1** | 1633.1 |
| S2_4i6s | 1505.07 | 6 | **1505.07** | 6 | **1505.07** | 1505.07 | 6 | **1505.07** | 1505.07 | 6 | **1505.07** | 1505.07 | 6 | **1505.07** | 1505.07 |
| S2_6i6s | 2431.33 | 8 | 2660.49 | 7 | **2431.33** | 2431.33 | 7 | **2431.33** | 2431.33 | 7 | **2431.33** | 2431.33 | 7 | **2431.33** | 2431.33 |
| S2_8i6s | 2158.35 | 7 | 2175.66 | 7 | **2158.35** | 2158.35 | 7 | 2175.65 | 2175.65 | 7 | **2158.35** | 2158.35 | 7 | **2158.35** | 2158.35 |
| S2_10i6s | 1585.46 | 5 | **1585.46** | 6 | **1585.46** | 1585.46 | 5 | **1585.46** | 1585.46 | 6 | **1585.46** | 1585.46 | 6 | **1585.46** | 1585.46 |
| S1_4i2s | 1582.2 | 6 | 1598.91 | 6 | **1582.2** | 1582.2 | 6 | **1582.2** | 1582.2 | 6 | **1582.2** | 1582.2 | 6 | **1582.2** | 1582.2 |
| S1_4i4s | 1460.09 | 5 | 1483.19 | 5 | **1460.09** | 1460.09 | 5 | **1460.09** | 1460.09 | 5 | **1460.09** | 1460.09 | 5 | **1460.09** | 1460.09 |
| S1_4i6s | 1397.27 | 5 | 1413.97 | 5 | **1397.27** | 1397.27 | 5 | **1397.27** | 1397.27 | 5 | **1397.27** | 1397.27 | 5 | **1397.27** | 1397.27 |
| S1_4i8s | 1397.27 | 6 | **1397.27** | 5 | **1397.27** | 1397.27 | 5 | **1397.27** | 1397.27 | 5 | **1397.27** | 1397.27 | 5 | **1397.27** | 1397.27 |
| S1_4i10s | 1396.02 | 5 | **1396.02** | 5 | **1396.02** | 1396.02 | 5 | **1396.02** | 1396.02 | 5 | **1396.02** | 1396.02 | 5 | **1396.02** | 1396.02 |
| S2_4i2s | 1059.35 | 4 | **1059.35** | 4 | **1059.35** | 1059.94 | 4 | **1059.35** | 1059.35 | 4 | **1059.35** | 1059.35 | 4 | **1059.35** | 1059.35 |
| S2_4i4s | 1446.08 | 5 | **1446.08** | 5 | **1446.08** | 1446.08 | 5 | **1446.08** | 1446.08 | 5 | **1446.08** | 1446.08 | 5 | **1446.08** | 1446.08 |
| S2_4i6s | 1434.14 | 5 | **1434.14** | 5 | **1434.14** | 1434.95 | 5 | **1434.14** | 1434.14 | 5 | **1434.14** | 1434.14 | 5 | **1434.14** | 1434.14 |
| S2_4i8s | 1434.14 | 5 | **1434.14** | 5 | **1434.14** | 1434.95 | 5 | **1434.14** | 1434.14 | 5 | **1434.14** | 1434.14 | 5 | **1434.14** | 1434.14 |
| S2_4i10s | 1434.13 | 5 | **1434.13** | 5 | **1434.13** | 1434.94 | 5 | **1434.13** | 1434.13 | 5 | **1434.13** | 1434.13 | 5 | **1434.13** | 1434.13 |
| Avg gap above BKS | | | 0.46 | | 0.00 | 0.00 | | 0.04 | 0.04 | | 0.00 | 0.00 | | 0.00 | 0.00 |
| Avg. time (min) | | | 0.02 | | 0.02 | | | 0.01 | | | 0.006 | | | 0.003 | |

Values in bold indicate the best known solutions.

TABLE I: Computational results on GVRP small instances

| instance | BKS | 48A | | GVNS/TS | | | ITS | | | MSLS | | | MSTS-SP | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | $v$ | best | $v$ | best | Avg | $v$ | best | Avg | $v$ | best | Avg | $v$ | best | Avg |
| 111c_21s | 4770.47 | 18 | 4960.6 | 17 | 4772.43 | 4801.42 | 18 | 4952.07 | 4985.22 | 17 | **4770.47** | 4775.86 | 17 | **4770.47** | 4771.22 |
| 111c_22s | 4767.21 | 18 | 4914.2 | 17 | 4778.29 | 4802.69 | 18 | 4977.07 | 4981.37 | 17 | **4767.21** | 4772.66 | 17 | **4767.21** | 4773.37 |
| 111c_24s | 4767.14 | 18 | 4952.9 | 17 | 4773.67 | 4778.62 | 18 | 4949.23 | 4956.56 | 17 | **4767.14** | 4768.48 | 17 | **4767.14** | 4770.56 |
| 111c_26s | 4767.14 | 18 | 4934.11 | 17 | 4768.2 | 4785.26 | 18 | 4962.5 | 4975.56 | 17 | **4767.14** | 4769.5 | 17 | **4767.14** | 4770.42 |
| 111c_28s | 4765.52 | 18 | 4971.93 | 17 | 4767.03 | 4779.81 | 18 | 4967.07 | 4970.88 | 17 | **4765.52** | 4767.97 | 17 | **4765.52** | 4769.03 |
| 200c_21s | 8757.14 | 32 | 9276.63 | 31 | 8848.71 | 8902.73 | 32 | 9279.66 | 9326.71 | 31 | 8766.04 | 8790.8 | 31 | **8757.14** | 8784.71 |
| 250c_21s | 10379.98 | 39 | 11007.98 | 37 | 10496.11 | 10548.74 | 38 | 10901.15 | 10912.13 | 37 | **10379.98** | 10414.45 | 37 | 10406.81 | 10452.13 |
| 300c_21s | 12202.49 | 46 | 12869.17 | 44 | 12386.09 | 12452.28 | 45 | 12891.92 | 12903.03 | 44 | **12202.49** | 12209.94 | 44 | 12243.95 | 12283.03 |
| 350c_21s | 13908.96 | 54 | 14954.83 | 50 | 14100.29 | 14180.61 | 51 | 14592.25 | 14623.75 | 50 | **13908.96** | 13929.89 | 50 | 13944.24 | 14001.57 |
| 400c_21s | 16398.13 | 61 | 17351.92 | 59 | 16695.89 | 16751.32 | 61 | 17490.09 | 17551.44 | 58 | **16398.13** | 16424.29 | 58 | 16444.12 | 16481.78 |
| 450c_21s | 17938.85 | 68 | 19215.38 | 65 | 18289.31 | 18332.08 | 67 | 19049.21 | 19236.14 | 64 | **17938.85** | 17973.93 | 64 | 18001.55 | 18037.24 |
| 500c_21s | 20207.81 | 76 | 21636.59 | 73 | 20562.18 | 20665.92 | 75 | 21375.35 | 21522.37 | 72 | **20207.81** | 20245.13 | 72 | 20250.47 | 20345.69 |
| Avg gap above BKS | | | 5.31 | | 0.92 | 1.35 | | 5.11 | 5.41 | | 0.01 | 0.15 | | 0.14 | 0.37 |
| Avg. time (min) | | | 157.03 | | 13.22 | | | 13.09 | | | 49.71 | | | 26.47 | |

Values in bold indicate the best known solutions.

TABLE II: Computational results on GVRP large instances