Transportation Research Part C 70 (2016) 113–128

Contents lists available at ScienceDirect

Transportation Research Part C

journal homepage: www.elsevier.com/locate/trc

A multi-space sampling heuristic for the green vehicle routing problem

Alejandro Montoya [a,b], Christelle Guéret [a], Jorge E. Mendoza [c], Juan G. Villegas [d,*]

[a] Université d'Angers, LARIS (EA 7315), 62 avenue Notre Dame du Lac, 49000 Angers, France
[b] Departamento de Ingeniería de Producción, Universidad EAFIT, Carrera 49 No. 7 Sur-50, Medellín, Colombia
[c] Université François-Rabelais de Tours, CNRS, LI EA 6300, OC ERL CNRS 6305, 64 avenue Jean Portalis, 37200 Tours, France
[d] Departamento de Ingeniería Industrial, Universidad de Antioquia, Calle 70 No. 52-21, Medellín, Colombia

ARTICLE INFO

Article history:
Received 12 June 2014
Received in revised form 4 September 2015
Accepted 17 September 2015
Available online 14 October 2015

Keywords:
Vehicle routing problem
Green vehicle routing problem
Hybrid heuristic
Matheuristic

ABSTRACT

The green vehicle routing problem (Green VRP) is an extension of the VRP in which routes are performed using alternative fuel vehicles (AFVs). AFVs have limited tank capacity, so routes may visit alternative fuel stations (AFSs) en-route. We propose a simple yet effective two-phase heuristic to tackle the Green VRP. In the first phase our heuristic builds a pool of routes via a set of randomized route-first cluster-second heuristics with an optimal AFSs insertion procedure. In the second phase our approach assembles a Green VRP solution by solving a set partitioning formulation over the columns (routes) stored in the pool. To test our approach, we performed experiments on a set of 52 instances from the literature. The results show that our heuristic is competitive with state-of-the-art methods. Our heuristic unveiled 8 new best-known solutions, matched another 40, and delivered solutions with an average gap of 0.14% for the 4 remaining instances.

1. Introduction

Social awareness about the high environmental impact of transportation, depletion of oil reserves, and the quest for energy security have driven governments around the globe to harden legislation on the use of *environmentally unfriendly vehicles* and to provide incentives for the adoption of *alternative fuel vehicles* (AFVs) (Pollet et al., 2012). In response to this new context, in the last lustrum, companies and government agencies have dramatically increased the use of AFVs in their operations (Nesterova et al., 2013). For instance, since 2011, La Poste operates at least 250 electric vehicles (EVs) and signed orders for an additional 10,000 (Kleindorfer et al., 2012). Similarly, in 2013 Coca-Cola deployed more than 30 AFVs in selected cities across the U.S. (Priselac, 2013). In 2011, UPS purchased 130 hybrid electric delivery trucks (HEVs) and added them to its fleet of 250 HEVs (UPS, 2008).

The use of AFVs in freight transportation leads to new optimization problems. One of these problems is the green vehicle routing problem (Green VRP),[1] introduced by Erdoğan and Miller-Hooks (2012). The Green VRP is an extension of the

* Corresponding author.
E-mail addresses: jmonto36@eafit.edu.co (A. Montoya), christelle.jussien-gueret@univ-angers.fr (C. Guéret), jorge.mendoza@univ-tours.fr (J.E. Mendoza), juan.villegas@udea.edu.co (J.G. Villegas).

[1] Although in the green vehicle routing problem literature the most popular acronym is G-VRP, we prefer to use Green VRP to avoid confusions with the already established and more-studied generalized vehicle routing problem (GVRP).

http://dx.doi.org/10.1016/j.trc.2015.09.009
0968-090X/

well-known vehicle routing problem, arising when a fleet of AFVs based at a central depot services a set of geographically spread customers. The special feature of this VRP comes from the limited range of AFVs. To ensure the feasible completion of trips, the AFVs may visit alternative fuel stations (AFSs) en-route to refill the tank or recharge the battery.

Formally, the Green VRP is defined on an undirected and complete graph $G = (V, E)$. The vertex set $V = \{0\} \cup I \cup F = \{0, 1, 2, \ldots, n + a\}$ is made up of a depot (vertex 0), a set of customers $I = \{1, 2, \ldots, n\}$, and a set $F = \{n + 1, n + 2, \ldots, n + a\}$ of a AFSs. It is assumed that the depot can also be used as a refueling station, and that all refueling stations can handle an unlimited number of vehicles. Each vertex $i \in V$ has a service time τ_i. If $i \in I$ then τ_i is the service time at the customer; and if $i \in F \cup \{0\}$ then τ_i is the refueling time, which is assumed to be constant. The set $E = \{(i, j) : i, j \in V, i \neq j\}$ corresponds to edges connecting vertices of V. Each edge (i, j) has two associated nonnegative attributes: a travel time t_{ij} and a distance d_{ij}. The travel speeds are assumed to be constant over the edges. In addition, there is no limit on the number of stops that can be made for refueling. When refueling occurs, it is assumed that the tank is filled to its maximum capacity. The customers are served using a fleet of homogeneous AFVs with tank capacity Q and consumption rate cr. The vehicle driving-range constraint is dictated by the fuel tank capacity and a tour duration constraint T_{max}.

In the Green VRP the objective is to find a set of routes of minimum total distance such that each customer is visited exactly once; the level of the tank when the vehicle arrives at any vertex is nonnegative; each route satisfies the maximum-duration limit; and each route starts and ends at the depot. Fig. 1 depicts a feasible solution to a Green VRP.

The Green VRP is an NP-hard problem. Indeed, Lenstra and Rinnooy Kan (1981) showed that the classical VRP is NP-hard. Since the VRP is a special case of the Green VRP, we can conclude that the Green VRP is also NP-hard. Moreover, recent studies show that commercial solvers cannot solve to optimality instances of 20 customers in reasonable computational times (Schneider et al., 2014). Therefore, to tackle industrial-scale Green VRP instances we need heuristic approaches.

For the solution of the Green VRP we present a multi-space sampling heuristic (MSH), which is a simple yet effective heuristic introduced by Mendoza and Villegas (2013). The idea behind MSH is to sample different solution representation spaces and then to assemble a solution with (parts of) the sampled elements. Implementations of MSH have delivered competitive results to complex routing problems such as the VRP with stochastic demands (Mendoza and Villegas, 2013), the VRP with stochastic travel and service times (Gómez et al., 2015), and the combined maintenance and routing problem (Fontecha et al., 2015). The algorithm is built out of two main components: a set of *sampling functions* and an *assembling procedure*. The sampling functions are randomized route-first, cluster-second heuristics. Using these heuristics MSH draws a sample from the TSP solution representation space and extracts from it a sample of the route representation space. Later, MSH uses the sampled routes to assemble a final solution. The assembling procedure is a set partitioning model that runs over the set of routes sampled in the first phase. To implement our MSH, we adapted the randomized route-first, cluster-second heuristics proposed in Mendoza and Villegas (2013) to the Green VRP. This adaptation is far from being trivial, since route feasibility in the Green VRP is difficult to assess. Indeed, a route that is *fuel-infeasible* (that is, infeasible for fuel autonomy) can be repaired in a number of ways by inserting one or more visits to AFSs. Therefore, extracting Green VRP routes from a giant TSP tour is much more complex than in problems previously tackled using MSH. Existing metaheuristics for the Green VRP rely on insertion-based heuristics and neighborhood schemes for repairing fuel-infeasible routes (Erdoğan and Miller-Hooks, 2012; Schneider et al., 2014, 2015; Felipe et al., 2014). In general, these strategies consider only one insertion at a time and therefore are exposed to myopic choices. We propose an optimal procedure based on a reformulation of the repair problem as a constrained shortest path problem.

The main contributions of this research are threefold: (i) we introduce an optimal repair procedure based on a constrained shortest path formulation that inserts refueling stations into Green VRP routes. One of the main advantages

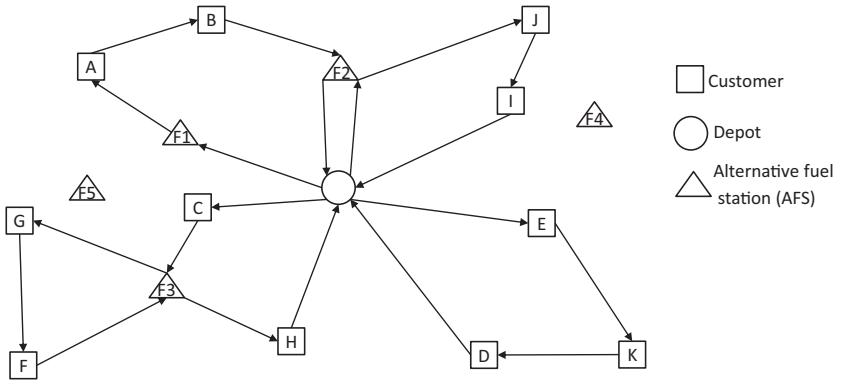


Fig. 1. Example of a feasible Green VRP solution.

of our procedure is that it can be used as a building block in any other Green VRP heuristic; (ii) we show how to use our procedure to build a simple and effective MSH heuristic for the Green VRP; and (iii) we update best-known solutions (BKSs) to 8 out of 52 standard benchmark instances for the problem.

The remainder of this paper is organized as follows. Section 2 reviews the related literature. Section 3 presents a detailed description of our version of the multi-space sampling heuristic. Section 4 presents a computational evaluation of the proposed method. Finally, Section 5 concludes the paper and outlines future work. For easy reference, Appendix A summarizes our notation.

2. Literature review

The Green VRP was introduced by Erdoğan and Miller-Hooks (2012). They propose a mixed-integer-linear programming (MILP) formulation and two heuristics. The first heuristic is a modified Clarke and Wright savings algorithm (MCWS) that repairs infeasible routes by inserting AFSs using a savings criterion, and removes redundant AFSs after merging the routes. The second heuristic is a density-based clustering algorithm (DBCA) that first builds clusters and then runs MCWS for each cluster. In their article, the authors propose two sets of test instances: the set of "small" instances has 40 test problems with 20 customers, and the set of "large" instances has 12 instances with 111 to 500 customers. Computational experiments showed that CPLEX 11.2 was unable to solve to optimality even small instances using the MILP model. They also indicated that there were no significant differences in performance between MCWS and DBCA.

The Green VRP is closely related to the classical distance-constrained VRP (Laporte et al., 1985), but the latter does not consider the possibility of extending the vehicle's distance limitation. Another problem that is closely related to the Green VRP is the multi-depot vehicle routing problem with inter-depot routes (MDVRPI) described by Crevier et al. (2007); this problem considers intermediate depots at which vehicles can be replenished with goods. To solve this problem Crevier et al. (2007) presented a heuristic procedure that combines ideas from adaptive memory programming, tabu search, and integer programming. Tarantilis et al. (2008) proposed a hybrid guided local search heuristic that outperforms the procedure proposed by Crevier et al. (2007). Muter et al. (2014) proposed a branch and price algorithm for the MDVRPI; it solves to optimality some instances with up to 50 customers.

Schneider et al. (2014) introduced the electric vehicle routing problem with time windows and recharging stations (E-VRPTW), which is an extension of the VRP with a fleet of EVs. The E-VRPTW considers limited vehicle freight capacities, customer time windows, and the possibility of recharging at any of the available stations using an appropriate recharging scheme. Schneider et al. (2014) presented an MILP formulation and a hybrid metaheuristic combining variable neighborhood search and tabu search (VNS/TS). Their VNS/TS explores infeasible solutions with respect to capacity, time windows, and battery-usage constraints. A dynamic penalizing scheme is used to guide the search toward feasible solutions. The VNS component explores 15 neighborhoods based on cyclic exchanges (Thompson and Psaraftis, 1993) that transfer between routes sequences of customers of arbitrary length. Furthermore, Schneider et al. (2014) improved the MILP formulation proposed by Erdoğan and Miller-Hooks (2012), and they evaluated their VNS/TS and MILP approaches on the 52-instance testbed proposed by Erdoğan and Miller-Hooks (2012). Computational experiments showed that CPLEX 12.2 was unable to solve to optimality instances with 20 customers using their MILP model, and VNS/TS outperformed the constructive heuristics proposed by Erdoğan and Miller-Hooks (2012).

Schneider et al. (2015) introduced the vehicle routing problem with intermediate stops (VRPIS) that generalizes the Green VRP and the MDVRPI. To solve the VRPIS Schneider et al. (2015) propose an adaptive variable neighborhood search (AVNS). Their AVNS uses a modified savings algorithm to generate an initial solution that is later improved with local search. The algorithm uses an adaptive shaking with twenty-four neighborhood structures, five route selection methods, three vertex-sequence selection methods, and an adaptive mechanism to choose the route and vertex-selection methods. The solution generated at the shaking step is subsequently improved by several greedy local searches. Furthermore, the AVNS has a dynamic penalization scheme to guide the search toward feasible solutions and a simulated annealing acceptance criterion. Since the Green VRP is a special case of the VRPIS, Schneider et al. (2015) tested their approach on the instances of Erdoğan and Miller-Hooks (2012). This method outperformed all previous methods both in terms of solution quality and computational time.

Recently, Felipe et al. (2014) introduced the green vehicle routing problem with multiple technology and partial recharges (GVRP-MTPR). As the name suggests, in this problem charging stations may have different technologies (e.g., charging times) and EVs do not necessarily charge their batteries to the full capacity when they reach a charging point. In their article, the authors presented an MILP formulation, a local search method (48A), and a simulated annealing (SA). Their local search method uses 48 possible combinations of 6 different neighborhoods and selects the best overall solution. Their SA uses a relocate neighborhood to explore the solution space. Every time that the incumbent solution is updated, the SA uses a deterministic local search to try to further improve the solution. Since the GVRP-MTPR is an extension of the Green VRP, Felipe et al. (2014) tested their approaches on the instances of Erdoğan and Miller-Hooks (2012). Their methods outperformed the constructive heuristics proposed by Erdoğan and Miller-Hooks (2012), but are not competitive with the solution approaches of Schneider et al. (2014) and Schneider

et al. (2015). A plausible explanation for these results, is that their 48A algorithm and their SA are not specifically tailored to the Green VRP.

Sassi et al. (2014) tackled an electric vehicle routing problem with heterogeneous mixed fleet and time dependent charging costs. In their problem the authors consider a number of realistic features such as: different charging technologies, coupling constraints between vehicles and charging technologies, charging station availability time windows, and charging costs depending on the time of the day. Regarding charging station capacity, the authors impose the maximum admissible power charge constraint over the charging spots located at the depot; however, they assume that the other charging stations have an unlimited capacity. They proposed an MILP formulation, a charging routing heuristic and a local search heuristic. They performed computational experiments on real data instances.

Gonçalves et al. (2011) considered a VRP with pick-up and delivery (VRPPD) with a mixed fleet that consists of battery electric vehicles (BEVs) and vehicles with internal-combustion engines. The objective is to minimize the total costs (vehicle-related fixed and variable costs). They considered time and capacity constraints and assumed a time for recharging the BEVs, which they calculated from the total distance traveled and the range for one battery charge. However, they did not incorporate the location of the recharging stations into their model. Thus, they basically proposed a mixed-fleet VRPPD with an additional distance-dependent time variable. They performed computational experiments with an MILP formulation on instances with up to 17 customers.

Conrad and Figliozzi (2011) introduced the recharging vehicle routing problem (RVRP) wherein vehicles with a limited range must service a set of customers but may recharge at certain customer locations before continuing their trip. They proposed an MILP formulation of the problem. They performed computational experiments on instances of 40 customers.

Juan et al. (2014) discussed the vehicle routing problem with multiple driving ranges (VRPMDR), an extension of the classical routing problem where the total distance each vehicle can travel is limited and is not necessarily the same for every vehicle. The VRPMDR finds applications in routing electric and hybrid-electric vehicles, which can cover limited distances depending on the running time of their batteries. They proposed an MILP formulation and a multi-round heuristic algorithm that iteratively constructs a solution for the problem.

Finally, Pelletier et al. (2014) presented an overview of the field of goods distribution with EVs, that includes a review of the main transportation science literature on EVs regarding fleet size and mix, vehicle routing problem, and optimal paths.

3. Multi-space sampling heuristic

Mendoza and Villegas (2013) originally proposed the multi-space sampling heuristic for the vehicle routing problem with stochastic demands. Despite its simple design, MSH obtains competitive results. MSH has two phases: sampling and assembling. In the sampling phase the algorithm uses a set of randomized TSP heuristics to draw a biased sample from the set $\mathcal{K}$ of TSP-like tours (i.e., giant tours visiting all customers). Following the route-first cluster-second principle (Beasley, 1983; Prins et al., 2014), MSH extracts every feasible route that can be obtained without altering the order of the customers of each sampled TSP tour. MSH uses these routes to build a set $\Omega \subset \mathcal{R}$, where $\mathcal{R}$ is the set of all feasible routes. In the assembling phase MSH follows the principle of petal heuristics (Foster and Ryan, 1976) and maps set Ω to a solution s (in the set of all feasible solutions to the problem $\mathcal{S}$) by solving a set partitioning formulation. Note that in MSH the knowledge of the problem is embedded in two components: the procedure that extracts routes from the TSP tours during the sampling phase and the set-partitioning formulation used in the assembling phase. The former controls the feasibility and cost of each route while the latter controls the feasibility and cost of the whole solution. To adapt MSH to the Green VRP we designed a tailored route extraction procedure. To favor scalability we also modified the strategy used by the original MSH to build the route pool Ω. For the sake of completeness, in the remainder of this section we present all the components used in our MSH setting a special focus on those we specifically designed for the Green VRP.

3.1. General structure

Algorithm 1 describes the general structure of our MSH. The procedure starts by entering the sampling phase (lines 4–19). At each iteration $k \leqslant K$, the algorithm selects a sampling heuristic from a set $\mathcal{H}$ (line 6) and uses it to build a TSP tour *tsp*. Then, the algorithm uses a tour splitting procedure (known as `split`) to retrieve a solution $s \in \mathcal{S}$. Differently to the original MSH, our version does not store in Ω all the routes evaluated by `split` during the partitioning process but only the routes belonging to some of the retrieved solutions. To decide if the routes of a solution s should join Ω we use the following condition (lines 9–16): $f(s) \leqslant f(s^*) \cdot (1 + \lambda)$ where s^* is the best solution found, λ a positive parameter, and $f(\cdot)$ denotes the objective function of a solution. The idea behind this choice is to favor computational scalability by reducing the size of Ω while assuring a good compromise between the diversity and the quality of the routes in the pool. In the assembly phase (line 20), the heuristic invokes a procedure called `SetPartitioning` to solve a set partitioning formulation over Ω using $f(s^*)$ as an upper bound. The resulting solution $\overline{\mathcal{R}}$ is reported by the heuristic (line 21).

Algorithm 1. Multi-space sampling heuristic: General structure.

```
1:  function MultiSpaceSampling(G, H, K, λ)
2:    Ω ← ∅
3:    k ← 1
4:    while k ⩽ K do
5:      for j = 1 to j = |H| do
6:        h ← H_j
7:        tsp ← h(G)
8:        s ← split(G, tsp)
9:        if k = 1 and j = 1 then
10:         s* ← s
11:       else if f(s) ⩽ f(s*) × (1 + λ) then
12:         Ω ← Ω ∪ s
13:         if f(s) < f(s*) then
14:           s* ← s
15:         end if
16:       end if
17:       k ← k + 1
18:     end for
19:   end while
20:   R̄ ← SetPartitioning(G, Ω, s*)
21:   return R̄
22: end function
```

3.2. Sampling heuristic

To sample $\mathcal{K}$ (line 7 in Algorithm 1), our approach uses randomized versions of three TSP constructive heuristics. Although the strategies used to generate the randomized versions of the three heuristics are directly borrowed from Mendoza and Villegas (2013), for the sake of completeness we briefly describe them here.

Let *tsp* be an ordered set representing the TSP customer tour being built by a given sampling heuristic, $\mathcal{W}$ the set of customers visited by *tsp*, and $\mathcal{N} = I \setminus \mathcal{W}$ an ordered set of nonrouted customers. For the sake of simplicity, we assume that sets $\mathcal{W}$ and $\mathcal{N}$ are updated every time a customer is added to *tsp*. Let l be a random integer in $\{1, \ldots, min\{L, |\mathcal{N}|\}\}$, where parameter L denotes the *randomization factor* of each heuristic.

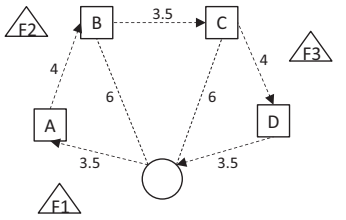
The first heuristic is randomized nearest neighbor (RNN). It initially sets $tsp = (0)$ and $u = 0$. At each iteration, RNN identifies the customer v that is the lth nearest customer to u, appends v to *tsp*, and sets $u = v$. RNN stops when $|\mathcal{N}| = 0$ and appends 0 to *tsp* to complete the tour. The second heuristic is randomized nearest insertion (RNI). It initializes *tsp* as a tour starting at the depot and performing a round trip to a randomly selected customer. At each iteration, RNI sorts $\mathcal{N}$ in nondecreasing order of $d_{min}(v)$, where $d_{min}(v)$ is defined to be $min\{d_{u,v} | u \in \mathcal{W}\}$. RNI then inserts $v = \mathcal{N}_l$ (i.e., the lth element in the ordered set $\mathcal{N}$) into the tour *tsp* in the best possible position (i.e., the position generating the smallest increment in the cost of the tour). RNI stops when $|\mathcal{N}| = 0$. The third heuristic is randomized best insertion (RBI). It initializes *tsp* as a tour starting at the depot and performing a round trip to a randomly selected customer. At each iteration RBI sorts $\mathcal{N}$ in nondecreasing order of $\Delta_{min}(v)$, where $\Delta_{min}(v)$ is defined to be $min\{d(u, v) + d(v, w) - d(u, w) | (u, w) \in tsp\}$, and inserts $v = \mathcal{N}_l$ in tour *tsp* in the best possible position. RBI stops when $|\mathcal{N}| = 0$.

3.3. Split

To extract a feasible solution s from *tsp* (line 8 in Algorithm 1), our approach uses an adaptation of the optimal tour splitting procedure for the VRP introduced by Prins (2004). The splitting procedure builds a directed acyclic graph $G^* = (V^*, A)$ composed of the ordered vertex set $V^* = (v_0, v_1, \ldots, v_i, \ldots, v_n)$ and the arc set A. Vertex $v_0 = 0$ is an auxiliary vertex, while vertices $v_1, \ldots, v_n \in tsp \setminus \{0\}$. The vertices in V^* are arranged in the same order in which they appear in *tsp*. Arc $(v_i, v_{i+n_r}) \in A$ represents a feasible route $r_{v_i, v_{i+n_r}}$ with distance $d_{r_{v_i, v_{i+n_r}}}$ starting and ending at the depot and visiting customers in the sequence v_{i+1} to v_{i+n_r}. Route $r_{v_i, v_{i+n_r}}$ may not satisfy the fuel constraint (i.e., the route's fuel consumption is greater than Q). If it does not, we try to repair it by inserting visits to AFSs. If the insertion of AFSs increases the duration of the route beyond T_{max} then we do not include the arc associated with the route in G^*. The insertion of visits to AFSs is accomplished by the optimal repair procedure explained in Section 3.4. To obtain a feasible solution s for the Green VRP, the procedure finds the set of arcs (i.e., routes) along the shortest path connecting 0 and v_n in G^*. It is worth noting that since G^* is directed and acyclic, building the graph and finding the shortest path can be done simultaneously (Prins, 2004).

Algorithm 2 shows the tour splitting procedure based on the split algorithm for the capacitated VRP (Prins, 2004). After initializing the shortest path labels (lines 2–6), split enters the outer loop (lines 7–41). Each pass through the outer loop sets the tail of an arc and initializes an empty route (lines 8–11). Then we use the inner loop (lines 13–40) to scan all the arcs sharing the same tail node. At each inner-loop iteration, we explore a new arc by simply adding the next customer to the route (line 16). In the next steps (lines 17–23) we compute the weight and time of the arc (i.e., the total distance c_r and total time tp_r of the associated route). If an arc is feasible for the duration constraint (line 24) we check the feasibility of the associated route with respect to the fuel constraint (and store the result in a boolean variable f_r). When a route is infeasible for the fuel constraint but $tp_r \leqslant T_{max}$, we try to repair the route using procedure `Repair(r,G)` (line 27). For an arc (i,j) to be added to the graph, its corresponding route must be feasible for fuel and time. If the shortest path label of the head node of arc (i,j) can be improved (i.e., $label_{i-1} + c_r \leqslant label_j$), then we update the shortest path label and the predecessor information (lines 35–38). The algorithm then moves to the next inner-loop iteration or exits the loop. After completing the outer loop we retrieve the solution using the tour *tsp* and the predecessor labels (for an algorithm to retrieve the solution we refer the reader to Prins (2004)).

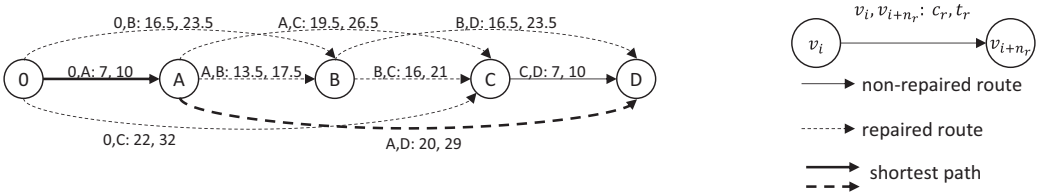
To adapt the tour splitting procedure to the Green VRP, it is necessary to introduce two important functions: `checkFuel` (line 25), which evaluates the feasibility of a route with respect to the vehicle's fuel constraint, and `Repair` (line 27), which tries to repair the route, and returns a boolean variable (indicating whether or not the route could be repaired) and the distance of the repaired route. Note, however, that there are some cases where the repaired route is feasible for the fuel constraint but infeasible for the duration constraint. This occurs because the time needed to recover the feasibility of the route



$Q = 7$
$T_{max} = 33$
$cr = 1$
$speed = 1$
$\tau_i = 1 \; \forall i \in F \cup \{0\}$
$\tau_i = 2 \; \forall i \in I$

Distance matrix between vertices and AFSs			
	F1	F2	F3
0	3.0	6.5	5.5
A	2.5	4.0	7.0
B	6.5	1.0	6.0
C	8.0	4.5	2.5
D	6.5	7.0	2.5

(a) TSP tour and instance data



(b) Auxiliary graph

Arc (route) evaluation						
Arc	Sequence	Feasible*		Repaired**	Distance	Time
		Tmax	Autonomy			
0,A	0, A, 0	✔	✔	N	7.0	10.0
0,B	0, F1, A, F2, B, 0	✔	✔	✔	16.5	23.5
0,C	0, F1, A, F2, B, C, F3, 0	✔	✔	✔	22.0	32.0
0,D	0, F1, A, F2, B, C, F3, D, 0	✘	✔	✘	22.5	34.5
A,B	0, B, F2, 0	✔	✔	✔	13.5	17.5
A,C	0, B, F2, C, F3, 0	✔	✔	✔	19.5	26.5
A,D	0, B, F2, C, F3, D, 0	✔	✔	✔	20.0	29.0
B,C	0, F3, C, F3, 0	✔	✔	✔	16.0	21.0
B,D	0, F3, C, F3, D, 0	✔	✔	✔	16.5	23.5
C,D	0, D, 0	✔	✔	N	7.0	10.0

* ✔: feasible, ✘: infeasible

** **N**: route does not need to be repaired ✔: route can be repaired, ✘: route can not be repaired

(c) Arc evaluation

Fig. 2. Splitting a TSP tour into a Green VRP solution.

(i.e., the travel time to the AFSs and the service time there) increases its planned duration beyond T_{max}; in these cases we do not add the associated arc to the graph.

Fig. 2 illustrates the tour splitting procedure. Fig. 2a shows the TSP tour and the relevant information. Fig. 2b depicts the auxiliary graph G^*; the arcs in bold correspond to the shortest path. Fig. 2c is a table showing the evaluation of the arcs.

Algorithm 2. Tour splitting.

```
1:  function SPLIT(G, tsp)
2:    label_0 ← 0                                   ▷ label: shortest path labels
3:    for i = 1 to n do
4:      label_i ← +∞
5:      pred_i ← 0                                  ▷ pred: predecessor labels
6:    end for
7:    for i = 1 to n do
8:      r ← (0)                                     ▷ Initialize route r
9:      c_r ← 0                                     ▷ Initialize total distance c_r
10:     tp_r ← 0                                    ▷ Initialize total time tp_r
11:     j ← i
12:     continue ← true
13:     while j ⩽ n and continue = true do
14:       v ← tsp_j                                 ▷ Get customer in position j of tsp
15:       v' ← tsp_{j−1}                            ▷ Get customer in position j − 1 of tsp
16:       r ← r ∪ {v}
17:       if j = i then
18:         c_r ← d_{0,v} + d_{v,0}                 ▷ Update total distance
19:         tp_r ← t_{0,v} + t_{v,0} + τ_v + τ_0    ▷ Update total time
20:       else
21:         c_r ← c_r − d_{v',0} + d_{v',v} + d_{v,0}   ▷ Update total distance
22:         tp_r ← tp_r − t_{v',0} + t_{v',v} + t_{v,0} + τ_v   ▷ Update total time
23:       end if
24:       if tp_r ⩽ T_max then                      ▷ Check feasibility for time
25:         f_r ← checkFuel(r)                      ▷ Evaluate fuel feasibility
26:         if f_r = false then
27:           ⟨c_r, f_r⟩ ← Repair(r, G)             ▷ Repair route r
28:           if f_r = false then
29:             continue ← false
30:           end if
31:         end if
32:       else
33:         continue ← false
34:       end if
35:       if label_{i−1} + c_r ⩽ label_j and continue = true then
36:         label_j ← label_{i−1} + c_r             ▷ Update label
37:         pred_j ← i − 1                          ▷ Update predecessor
38:       end if
39:       j ← j + 1
40:     end while
41:   end for
42:   s ← RetrieveSolution(pred, tsp)
43:   return s
44: end function
```

3.4. Repair procedure

Existing approaches for the Green VRP depend on insertion-based heuristics and neighborhood schemes for repairing fuel-infeasible routes. The MCWS algorithm proposed by Erdoğan and Miller-Hooks (2012) starts inserting the AFS with the least insertion cost (i.e., distance added to the tour) in the fuel-infeasible back-and-forth routes. After the routes merging step, MCWS inserts into each fuel-infeasible route the AFS with the least insertion cost and then evaluates the possibility of

eliminating redundant AFSs. Schneider et al. (2014) used a penalization scheme for the fuel-infeasible routes and proposed an operator that performs insertions and removals of AFSs. That operator uses a tabu list in order to control insertions already tested. Schneider et al. (2015) used a penalization scheme for the fuel-infeasible routes and presented three operators to improve the location of AFSs within the route. The first operator moves an AFS to a different position within a route. The second operator evaluates for each AFS visit of each route whether visiting a different AFS decreases the routing cost. The third operator aims at removing redundant AFS visits. Finally, Felipe et al. (2014) proposed a constructive heuristic, which considers the insertion of AFSs to fuel-infeasible routes. They presented a neigboorhood scheme to improve the location of an AFS in a route, without modifying the sequence of visits to the customers. Notice that all these strategies consider only one insertion at a time and therefore are exposed to myopic choices. In contrast, our repair procedure optimally chooses which stations to insert and where to insert them while considering all the possible combinations of insertions leading to an energy-feasible route. To accomplish its goal, our procedure relies on a reformulation of the repair problem as a constraint shortest path problem.

Let $\Pi = \{\pi_1, \ldots, \pi_i, \ldots, \pi_j, \ldots, \pi_{n_r+1}\}$ be a route that violates the fuel constraint, and $\pi_1 = 0$. The feasibility of Π may be restored by inserting visits to AFSs. We use an optimal procedure that simultaneously decides which stations must be visited and their optimal insertion position within the route. The procedure can be seen as a constrained shortest path problem in a repair graph $B = (Z, U)$. Fig. 3 depicts the structure of B.[2] The vertex set $Z = \{\alpha, \ldots, [i,k], \ldots, \beta\}$ is made up of two dummy vertices (α and β) that act as copies of the depot, representing the source and sink vertices of B; and the vertices $[i,k]$, which represent a visit to station $k \in F$ after visiting vertex π_i (i.e., the ith element in route Π), where $1 \leqslant i \leqslant n_r + 1$ and $1 \leqslant k \leqslant a$. To define the edge set U, let us first introduce some key elements. Let $P = \left\{p_1, \ldots, p_j, \ldots, p_{|P|}\right\}$ be a path in G. For a given path P we define three metrics: its distance $d(P) = \sum_{j=1}^{|P|-1} d_{p_j p_{j+1}}$, its total planned time $t(P) = \sum_{j=1}^{|P|-1} t_{p_j p_{j+1}} + \sum_{j=2}^{|P|} \tau_{p_j}$, and its fuel consumption $q(P) = cr \times d(P)$. The arc set U is composed of five types of arcs, that is, $U = \bigcup_{i=1}^{5} U_i$, where:

- U_1, the outgoing arcs of α. An arc $(\alpha, [i,k])$ represents the path $P = (0, F_k)$ if $i = 1$, and $P = (0, \ldots, \pi_i, F_k)$ if $i > 1$. Its cost and time are defined as $\bar{c}_{u:u\in U_1} = d(P)$ and $\bar{t}_{u:u\in U_1} = t(P) + \tau_0$.
- U_2, the arcs connecting two stations without visiting any customer. Arc $([i,k],[i,l])$ represents the path $P = (F_k, F_l)$. Its cost and time are defined as $\bar{c}_{u:u\in U_2} = d(P)$ and $\bar{t}_{u:u\in U_2} = t(P)$.
- U_3, the arcs connecting two stations and visiting some customers in between. Arc $([i,k],[j,l])$ represents the path $P = (F_k, \pi_{i+1}, \ldots, \pi_j, F_l)$. Its cost and time are defined as $\bar{c}_{u:u\in U_3} = d(P)$ and $\bar{t}_{u:u\in U_3} = t(P)$.
- U_4, the incoming arcs to β representing the return to the depot after visiting some customers since the last visit to a station. Arc $([i,k],\beta)$ represents the path $P = (F_k, \pi_{i+1}, \ldots, \pi_{n_r+1}, 0)$. Its cost and time are defined as $\bar{c}_{u:u\in U_4} = d(P)$ and $\bar{t}_{u:u\in U_4} = t(P) - \tau_0$.
- U_5, the incoming arcs to β representing a return to the depot directly from a station after visiting the last customer in route Π. The arc $([n_r+1,k],\beta)$ represents the path $P = (F_k, 0)$. Its cost and time are defined as $\bar{c}_{u:u\in U_5} = d(P)$ and $\bar{t}_{u:u\in U_5} = t(P) - \tau_0$.

We include in U only arcs with $\bar{c}_u \times cr \leqslant Q$ and $\bar{t}_u \leqslant T_{max}$. With this graph construction procedure we ensure that all paths from α to β represent a route that visits customers in the same order they appear in Π and is feasible with respect to the fuel constraint. Note, however, that not every path connecting α and β in B represents a route that is feasible in terms of the duration constraint. Therefore, to find an optimal repair for route Π, we need to solve a constrained shortest path problem (CSP), where the maximum travel time T_{max} is the constrained resource. It is interesting to observe that since all the arcs of B are feasible for fuel, the tank capacity is not a constraint for the CSP. To solve the CSP we use the pulse algorithm (Lozano et al., 2013). The algorithm is based on the idea of propagating pulses through a network from a source node to a sink node (α and β in our case). At the core of the algorithm lies the ability to effectively and aggressively prune pulses (i.e., prevent their propagation) without jeopardizing the optimal path. The pulse algorithm is one of the state-of-the-art methods for the solution of resource-constrained shortest path problems (Lozano and Medaglia, 2013).

Fig. 4 shows a detailed example of the repair procedure. Fig. 4a illustrates an infeasible route with three customers and three AFSs for a Green VRP. Fig. 4b shows the route after the optimal insertion of the AFSs using the repair procedure. Fig. 4c shows the corresponding repair graph and the shortest path.

3.5. Set partitioning

In the assembly phase, MSH maps the set Ω to a solution in S by solving a set partitioning formulation $\left(min_{\overline{R}\subseteq\Omega}\{\sum_{r\in\overline{R}} d_r : \cup_{r\in\overline{R}} = V; r_i \cap r_j = 0\ \forall r_i, r_j \in \overline{R}\}\right)$. The objective is then to select the best subset of routes from Ω to build the set of routes $\overline{R}$ (i.e., the final solution) guaranteeing that each customer will be visited by exactly one route.

[2] This graph resembles that used by Villegas et al. (2010) in a route-first cluster-second heuristic for the single truck and trailer routing problem with satellite depots.

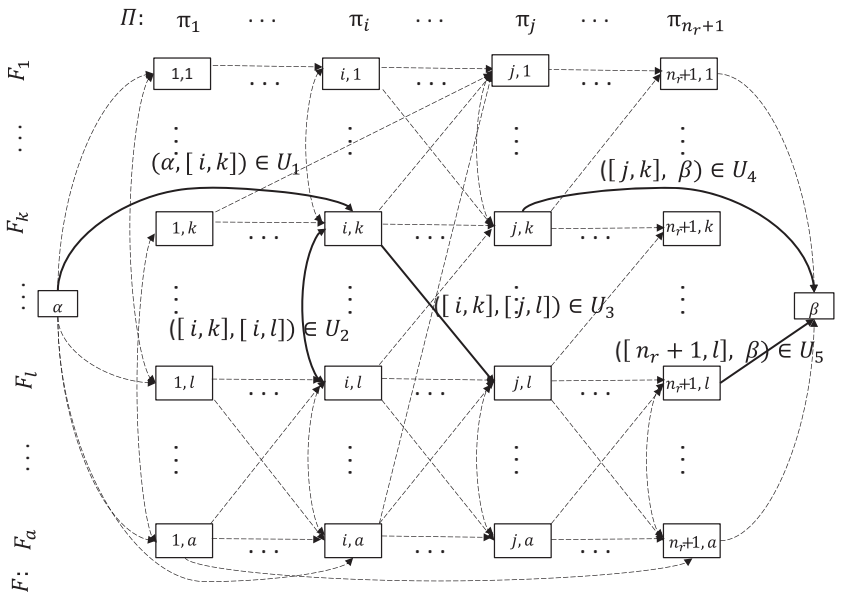


Fig. 3. Outline of the structure of the repair graph $B = (Z, U)$.

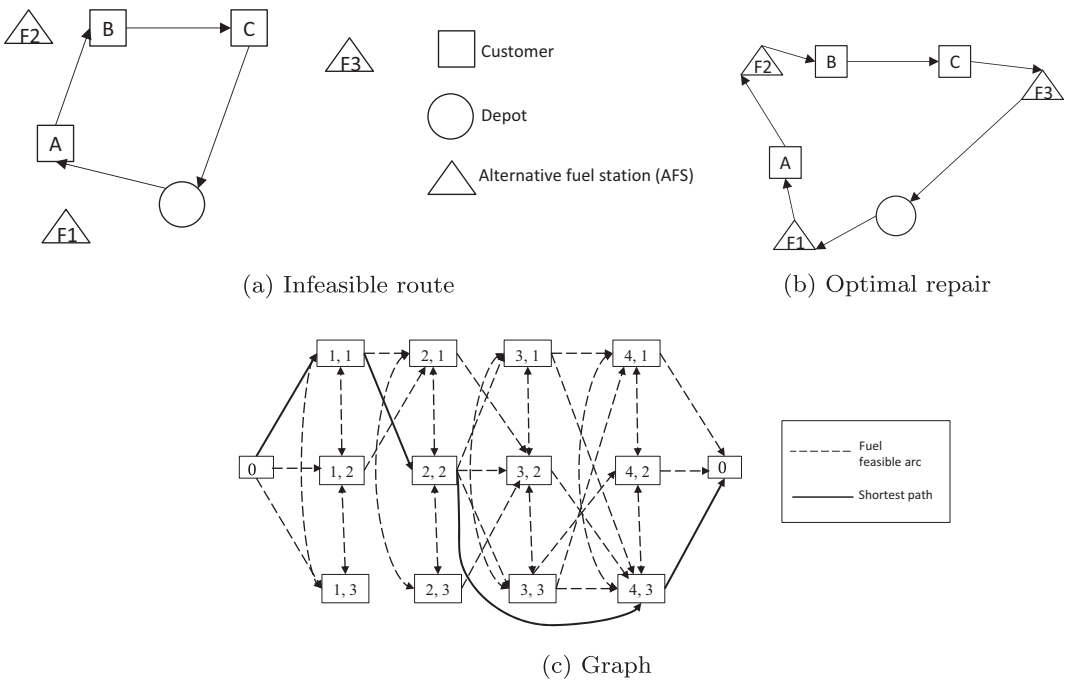


Fig. 4. Optimal repair example for the three-customer sequence $\Pi = \{0, A, B, C, 0\}$, three AFSs.

4. Computational experiments

To test our approach, we ran it on the 52-instance testbed proposed by Erdoğan and Miller-Hooks (2012). These instances consist of 5 sets. Four sets contain ten 20-customer instances ranging from 2 to 10 AFSs. The remaining set, resulting from a case study, consists of 12 instances with the number of customers ranging from 111 to 500 and between 21 and 28 AFSs. In this testbed there are instances with infeasible customers. Therefore, Erdoğan and Miller-Hooks (2012), Schneider et al. (2014, 2015), Felipe et al. (2014) filter the customers that either cannot be served directly within the maximum route duration or whose direct service requires visiting more than one refueling station. To allow comparison with previously

Table 1
Summary results and comparison of our MSH with other methods on the small instances of Erdoğan and Miller-Hooks (2012).

Metric	MCWS/DBCA	VNS/TS	AVNS	48A	MSH(1k)	MSH(5k)	MSH(10k)
Number of BKS	2/40	38/40	40/40	29/40	35/40	40/40	40/40
Avg. gap (%)	NR	NR	0.15	NR	0.38	0.04	0.01
Max. gap (%)	NR	NR	1.73	NR	3.61	0.47	0.13
Avg. best gap (%)	8.72	0.63	0.00	0.46	0.09	0.00	0.00
Cum. number of veh.	245	223	NR	225	222	222	222
Avg. time (min)	NR	0.65	0.17	0.02	0.01	0.04	0.07
Max. avg. time (min)	NR	0.88	0.38	0.04	0.02	0.06	0.12
Computer	Pentium 4 3.2 GHz	Core I5 2.67 GHz	Core I5 2.67 GHz	Core I5 2.8 GHz		XEON E5410 2.33 GHz	
Runs	NR	10	10	1	10	10	10

NR: not reported.

Table 2
Summary results and comparison of our MSH with other methods on the large instances of Erdoğan and Miller-Hooks (2012).

Metric	MCWS/DBCA	VNS/TS	AVNS	48A	SA	MSH (1k)	MSH (5k)	MSH (10k)
Number of BKS	0/12	0/12	4/12	0/12	0/12	0/12	0/12	8/12
Avg. gap (%)	NR	NR	0.92	NR	NR	2.64	1.48	1.02
Max. gap (%)	NR	NR	1.84	NR	NR	5.77	4.21	3.62
Avg. best gap (%)	15.97	1.38	0.17	4.50	4.97	1.42	0.40	0.05
Cum. number of veh.	508	461	NR	466	459	454	445	444
Avg. time (min)	NR	159.58	6.20	157.03	156.05	27.92	31.47	35.04
Max. avg. time (min)	NR	525.52	19.51	514.68	456.26	80.11	84.95	89.95
Computer	Pentium 4 3.2 GHz	Core I5 2.67 GHz	Core I5 2.67 GHz	Core I5 2.8 GHz	Core I5 2.8 GHz		XEON E5410 2.33 GHz	
Runs	NR	10	10	1	1	10	10	10

NR: not reported.

published results, we followed the same convention and filter unfeasible customers. It is important to remark that in the problem definition an AFS vertex could have as successor another AFS, so it is possible to insert visits to more than one AFS between two customer visits. According to Erdoğan and Miller-Hooks (2012), the geographical coordinates given for the customers have to be converted to Cartesian coordinates using the Haversine formula (Bullard and Kiernan, 1992) with an average Earth radius of 4,182.45 miles. It is worth noting that although Erdoğan and Miller-Hooks (2012) include an additional constraint on the number of vehicles in their Green VRP formulation, their experiments are conducted with an unlimited-fleet version of the problem. Therefore, our results are directly comparable to theirs.

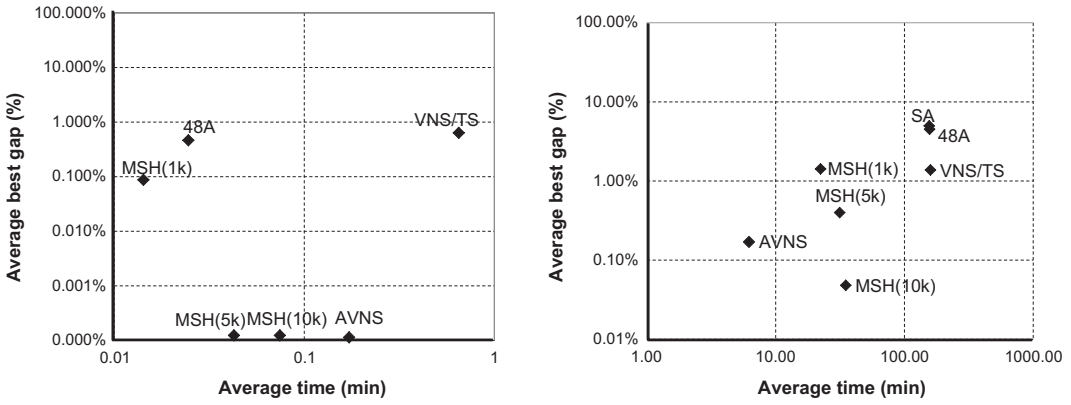
We implemented our MSH in Java (jre V.1.7.0_51) and used Gurobi Optimizer (version 5.6.0) to solve the set partitioning problem. We set a time limit of $10 \cdot n$ seconds on Gurobi to control the running time of the set partitioning problem. All the experiments were run on a computing cluster with 2.33 GHz Inter Xeon E5410 processors with 16 GB of RAM running under Linux Rocks 6.1.1. Each replication of the experiments was run on a single processor.

4.1. Results

After conducting a parameter tuning campaign, we set L_{RNN}, L_{RNI}, and L_{RBI} to 2, and $\lambda = 1$. We found that these parameters lead to a pool of well-diversified routes. For the sake of brevity, we will not discuss these experiments.

Tables 1 and 2 summarize the results delivered by our MSH on the small and large Green VRP instances running with 3 configurations: $K = 1,000; K = 5,000$; and $K = 10,000$ (defined as MSH(K)). We compare our results to the best result obtained by MCWS and DBCA by Erdoğan and Miller-Hooks (2012), the VNS/TS by Schneider et al. (2014), the AVNS by Schneider et al. (2015), and the 48A and SA by Felipe et al. (2014). The rows of Tables 1 and 2 indicate the number of times that each method found the BKS, the average and maximum gap between the average solution and the BKS (in %), the average best gap[3] (in %), the cumulative number of vehicles,[4] and the average and maximum computational times (in minutes). The results of VNS/TS, AVNS and our MSH are computed over 10 runs and the 48A and SA results over a single run. Erdoğan and Miller-Hooks (2012) reported the best solution of several runs with different parameters but did not give the exact number of runs. The detailed results are reported in Appendix B.

[3] The best gap is the gap between the best solution and the BKS.
[4] This metric is the sum of the number of vehicles of all best solutions (Bräysy and Gendreau, 2005).



(a) Small instances (%) (b) Large instances

Fig. 5. Trade-off between solution quality and CPU time.

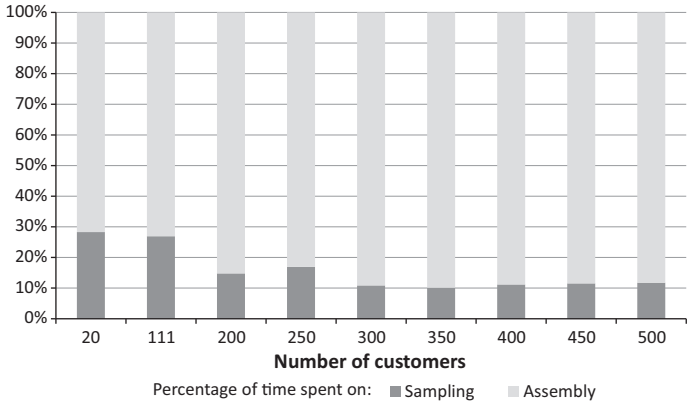


Fig. 6. Percentage share of CPU time by MSH phases.

For small instances, MSH(5k) and MSH(10k) have competitive results with reference to MCWS/ DBCA, VNS/TS, AVNS, and 48A (Table 1). They have an average gap of only 0.04% and 0.01%, match the 40 BKSs, and use the lowest cumulative number of vehicles. In terms of CPU time, MSH(5k) and MSH(10k) are only outperformed by MSH(1k) and 48A. MSH(1k) is the fastest method, and outperforms MCWS/DBCA, VNS/TS, and 48A in terms of solution quality.

For large instances, MSH(5k) outperforms MCWSA/DCBA, VNS/TS, 48A, and SA in terms of both solution quality (i.e., average best gap, and cumulative number of vehicles) and CPU time (Table 2). When compared to AVNS, MSH(10k) has a better average best gap with respect to BKSs (0.05% vs. 0.17%) and finds more BKSs (8/12 vs. 4/12), whereas AVNS is faster and it seems to scale better.

It is interesting to observe that for both the small and large instances our three MSH configurations use the lowest cumulative number of vehicles.

In order to give a graphical view of key metrics presented in Tables 1 and 2, Fig. 5 presents a head-to-head comparison between the solution methods. It shows the trade-off between solution quality and CPU time.[5] Each method is represented as a point in the plot. The *X* coordinate of the point represents the average best gap with respect to the BKS, while the *Y* coordinate represents the average CPU time. Notice that MCWS and DBCA were not included in the comparison since CPU times for these algorithms were not reported in by Erdoğan and Miller-Hooks (2012).

For the small instances, Fig. 5a shows that MSH(5k) and MSH(10k) dominate the AVNS and VNS/TS, while MSH(1k) dominates 48A and VNS/TS. Moreover, MSH(5k) has a remarkable performance on small instances, because its CPU time is close to that of 48A (which is the fastest approach in the literature) and its average best gap is 0.00%. On the other hand, on the large instances, MSH(5k) and MSH(10k) dominates SA, 48A and VNS/TS (Fig. 5b). The comparison between MSH(10k) and

[5] This comparison tool was introduced by Vidal et al. (2015).

AVNS shows that there is not clear dominator: while the former dominates in solution quality, the latter dominates in CPU time. Nonetheless, it is worth recalling that while AVNS uses elaborate components (i.e., an adaptive shaking with twenty-four neighborhood structures, five route selection methods, among others), our MSH uses simple building blocks that, with the notable exception of the repair procedure, are common in the literature. Therefore our MSH is probably easier to implement and extend to tackle other problems.

Finally, we analyzed the CPU time of the two MSH phases (sampling and assembly). Fig. 6 presents the percentage of time spent on each phase for each instance size (in terms of number of customers). The results show that, for all instances, assembly is the more time consuming phase, and that the percentage of time spent on the assembly phase increases with the instance size. However, the assembly phase contributes to improve the solutions on average by 4.10%, which is key to obtain competitive results with respect to existing approaches in the literature.

5. Conclusion and future work

This paper proposes a multi-space sampling heuristic for the Green VRP. This approach has three main components: a set of three randomized TSP heuristics, a tour partitioning procedure, and a set partitioning formulation. At the core of the tour partitioning procedure lies a novel repair mechanism that optimally inserts visits to refueling stations to restore the feasibility of routes violating the vehicle's fuel constraint. This procedure solves a constrained shortest path problem on an auxiliary repair graph, that by construction includes only the trips between refueling stations that respect the vehicle's fuel constraint.

The procedure is based on a reformulation of the repair problem as a constrained shortest path problem where (i) the maximum travel time of a route is the constrained resource, and (ii) each path from the source node to the sink node in the underlying graph models a possible way to repair the route by inserting visits to refueling stations.

We tested our approach on a 52-instance public testbed for the Green VRP. Our approach found 8 new BKSs for the testbed and matched another 40. When compared to state-of-the-art metaheuristics, the multi-space sampling heuristic obtains competitive results in terms of solution quality and computational time, and it is one of the simplest methods used to solve the Green VRP.

Perspectives for improving the computational performance of our approach include: (i) implementing faster methods for solving the set partitioning problem in the assembly phase (e.g., Lagrangian relaxation or dual ascent methods) and (ii) local searching the solutions retrieved by the splitting procedure and adding to the route pool only the routes found in the local optima (thus reducing the size of the problem to solve in the assembly phase). Besides, in ongoing research we are extending the problem definition to consider partial refueling and a particular characteristic of electric vehicles: the non-linear charging (refueling) functions.

Acknowledgement

The authors would like to thank the APOLO team (APOLO is the supercomputer of the center of scientific computing, Universidad EAFIT, Medellín, Colombia) for their support in the computational experiments, and Andrés L. Medaglia, Leonardo Lozano and Daniel Duque at the Universidad de Los Andes for allowing us to use their implementation of the pulse algorithm. This work was partly funded by Universidad EAFIT (Colombia), Programa Enlaza Mundos (Alcaldía de Medellín – Colombia), Programa de Movilidad Doctoral hacia Francia de Colfuturo – Embajada de Francia – ASCUN – Colciencias – Ministerio de Educación (Colombia), the Region Pays de la Loire (France), EMC2, and NOVALOG via project OLASI, and Universidad de Antioquia through Project MDC-11-01-05.

Appendix A. Nomenclature

A.1. Notation for problem definition

G	Green VRP underlying graph $G = (V, E)$
V	Set of vertices of G
0	Depot
I	Subset of V representing customers
F	Subset of V representing alternative fuel stations
E	Set of edges in G
d_{ij}	Distance between vertices i and j ($i, j \in V$)
t_{ij}	Travel time between vertices i and j ($i, j \in V$)
n	Number of customers
a	Number of alternative fuel stations
T_{max}	Tour duration constraint
cr	Fuel consumption rate
τ_i	Service time at a vertex. If $i \in I$ then τ_i is service time at customer; if $i \in F \cup \{0\}$ then τ_i is refueling time at AFS
Q	Tank capacity

Table B.3
Results of MSH on small instances of Erdoğan and Miller-Hooks (2012).

Instance	BKS	MCWS/DBCA			VNS/TS				AVNS				48A				MSH(1k)					MSH(5k)					MSH(10k)				
		n	*v*	Best	*n*	*v*	Best	*t*	*n*	Best	Avg.	*t*	*n*	*v*	Best	*t*	*n*	*v*	Best	Avg.	*t*	*n*	*v*	Best	Avg.	*t*	*n*	*v*	Best	Avg.	*t*
20c3sU1	1797.49	20	6	1797.51	20	6	**1797.49**	0.69	20	**1797.49**	1797.49	0.16	20	6	1805.41	0.03	20	6	**1797.49**	1797.88	0.01	20	6	**1797.49**	1797.49	0.04	20	6	**1797.49**	1797.49	0.08
20c3sU2	1574.77	20	6	1613.53	20	6	**1574.77**	0.64	20	**1574.78**	1574.78	0.15	20	6	**1574.78**	0.02	20	6	**1574.78**	1574.78	0.01	20	6	**1574.78**	1574.78	0.04	20	6	**1574.78**	1574.78	0.07
20c3sU3	1704.48	20	7	1964.57	20	6	**1704.48**	0.64	20	**1704.48**	1704.48	0.13	20	6	**1704.48**	0.02	20	6	**1704.48**	1706.08	0.01	20	6	**1704.48**	1704.48	0.04	20	6	**1704.48**	1704.48	0.07
20c3sU4	1482.00	20	6	1487.15	20	5	**1482.00**	0.65	20	**1482.00**	1482.00	0.17	20	5	**1482.00**	0.03	20	5	**1482.00**	1482.00	0.01	20	5	**1482.00**	1482.00	0.04	20	5	**1482.00**	1482.00	0.07
20c3sU5	1689.37	20	5	1752.73	20	6	**1689.37**	0.67	20	**1689.37**	1689.37	0.18	20	6	**1689.37**	0.04	20	6	**1689.37**	1689.37	0.01	20	6	**1689.37**	1689.37	0.04	20	6	**1689.37**	1689.37	0.07
20c3sU6	1618.65	20	6	1668.16	20	6	**1618.65**	0.67	20	**1618.65**	1618.65	0.15	20	6	**1618.65**	0.03	20	6	**1618.65**	1618.65	0.01	20	6	**1618.65**	1618.65	0.04	20	6	**1618.65**	1618.65	0.07
20c3sU7	1713.66	20	6	1730.45	20	6	**1713.66**	0.64	20	**1713.66**	1713.66	0.19	20	6	**1713.67**	0.03	20	6	**1713.66**	1714.95	0.01	20	6	**1713.66**	1714.03	0.04	20	6	**1713.66**	1713.87	0.07
20c3sU8	1706.50	20	6	1718.67	20	6	**1706.50**	0.67	20	**1706.50**	1706.50	0.16	20	6	1722.78	0.03	20	6	**1706.50**	1706.50	0.01	20	6	**1706.50**	1706.50	0.04	20	6	**1706.50**	1706.50	0.07
20c3sU9	1708.81	20	6	1714.43	20	6	**1708.81**	0.66	20	**1708.82**	1708.82	0.19	20	6	**1708.82**	0.04	20	6	1710.90	1711.14	0.01	20	6	**1708.82**	1710.09	0.04	20	6	**1708.82**	1709.65	0.07
20c3sU10	1181.31	20	5	1309.52	20	4	**1181.31**	0.64	20	**1181.31**	1181.31	0.23	20	4	**1181.31**	0.02	20	4	**1181.31**	1181.31	0.01	20	4	**1181.31**	1181.31	0.04	20	4	**1181.31**	1181.31	0.07
20c3sC1	1173.57	20	5	1300.62	20	4	**1173.57**	0.62	20	**1173.57**	1173.57	0.38	20	4	1178.97	0.03	20	4	**1173.57**	1173.57	0.01	20	4	**1173.57**	1173.57	0.04	20	4	**1173.57**	1173.57	0.07
20c3sC2	1539.97	19	5	1553.53	19	5	**1539.97**	0.58	19	**1539.97**	1539.97	0.21	19	5	**1539.97**	0.02	19	5	**1539.97**	1539.97	0.01	19	5	**1539.97**	1539.97	0.04	19	5	**1539.97**	1539.97	0.08
20c3sC3	880.20	12	4	1083.12	12	3	**880.20**	0.25	12	**880.20**	880.20	0.15	12	3	**880.20**	0.01	12	3	**880.20**	880.20	0.01	12	3	**880.20**	880.20	0.03	12	3	**880.20**	880.20	0.04
20c3sC4	1059.35	18	5	1091.78	18	4	**1059.35**	0.53	18	**1059.35**	1077.71	0.23	18	4	**1059.35**	0.02	18	4	**1059.35**	1088.54	0.01	18	4	**1059.35**	1059.94	0.04	18	4	**1059.35**	1059.94	0.06
20c3sC5	2156.01	19	7	2190.68	19	7	**2156.01**	0.60	19	**2156.01**	2156.01	0.14	19	7	**2156.01**	0.02	19	7	**2156.01**	2157.31	0.01	19	7	**2156.01**	2156.56	0.05	19	7	**2156.01**	2156.04	0.10
20c3sC6	2758.17	17	9	2883.71	17	8	**2758.17**	0.71	17	**2758.17**	2758.17	0.14	17	8	**2758.17**	0.02	17	8	**2758.17**	2758.17	0.01	17	8	**2758.17**	2758.17	0.04	17	8	**2758.17**	2758.17	0.08
20c3sC7	1393.99	6	5	1701.40	6	4	**1393.99**	0.18	6	**1393.99**	1393.99	0.04	6	4	**1393.99**	0.00	6	4	**1393.99**	1393.99	0.01	6	4	**1393.99**	1393.99	0.03	6	4	**1393.99**	1393.99	0.06
20c3sC8	3139.72	18	10	3319.74	18	9	**3139.72**	0.62	18	**3139.72**	3139.72	0.08	18	9	**3139.72**	0.02	18	9	**3139.72**	3139.72	0.01	18	9	**3139.72**	3139.72	0.05	18	9	**3139.72**	3139.72	0.12
20c3sC9	1799.94	19	6	1811.05	19	6	**1799.94**	0.60	19	**1799.94**	1799.94	0.16	19	6	**1799.94**	0.02	19	6	**1799.94**	1799.94	0.01	19	6	**1799.94**	1799.94	0.05	19	6	**1799.94**	1799.94	0.10
20c3sC10	2583.42	15	8	2644.11	15	8	**2583.42**	0.45	15	**2583.42**	2600.39	0.09	15	8	**2583.42**	0.02	15	8	**2583.42**	2583.42	0.01	15	8	**2583.42**	2583.42	0.04	15	8	**2583.42**	2583.42	0.07
S1_2i6s	1578.12	20	6	1614.15	20	6	**1578.12**	0.71	20	**1578.12**	1578.12	0.16	20	6	**1578.12**	0.03	20	6	**1578.12**	1578.12	0.01	20	6	**1578.12**	1578.12	0.04	20	6	**1578.12**	1578.12	0.07
S1_4i6s	1397.27	20	5	1541.46	20	5	**1397.27**	0.75	20	**1397.27**	1397.27	0.16	20	5	1413.97	0.03	20	5	**1397.27**	1397.27	0.01	20	5	**1397.27**	1397.27	0.04	20	5	**1397.27**	1397.27	0.07
S1_6i6s	1560.49	20	6	1616.20	20	5	**1560.49**	0.73	20	**1560.49**	1560.49	0.20	20	6	1571.30	0.03	20	5	**1560.49**	1563.95	0.01	20	5	**1560.49**	1561.19	0.04	20	5	**1560.49**	1560.49	0.07
S1_8i6s	1692.32	20	6	1882.54	20	6	**1692.32**	0.74	20	**1692.32**	1692.32	0.17	20	6	**1692.33**	0.03	20	6	**1692.32**	1692.32	0.01	20	6	**1692.32**	1692.32	0.04	20	6	**1692.32**	1692.32	0.07
S1_10i6s	1173.48	20	5	1309.52	20	4	**1173.48**	0.71	20	**1173.48**	1173.48	0.24	20	4	**1173.48**	0.03	20	4	**1173.48**	1173.48	0.01	20	4	**1173.48**	1173.48	0.04	20	4	**1173.48**	1173.48	0.07
S2_2i6s	1633.10	20	6	1645.80	20	6	**1633.10**	0.75	20	**1633.10**	1633.10	0.19	20	6	1645.80	0.03	20	6	**1633.10**	1633.10	0.01	20	6	**1633.10**	1633.10	0.05	20	6	**1633.10**	1633.10	0.09
S2_4i6s	1505.07	19	6	1505.07	19	5	1532.96	0.88	19	**1505.07**	1505.07	0.14	19	6	**1505.07**	0.02	19	6	**1505.07**	1505.07	0.02	19	6	**1505.07**	1505.07	0.05	19	6	**1505.07**	1505.07	0.09
S2_6i6s	2431.33	20	10	3115.10	20	7	**2431.33**	0.78	20	**2431.33**	2431.33	0.13	20	8	2660.49	0.04	20	7	**2431.33**	2439.92	0.02	20	7	**2431.33**	2431.33	0.04	20	7	**2431.33**	2431.33	0.07
S2_8i6s	2158.35	16	9	2722.55	16	7	**2158.35**	0.57	16	**2158.35**	2158.35	0.09	16	7	2175.66	0.02	16	7	**2158.35**	2158.35	0.01	16	7	**2158.35**	2158.35	0.03	16	7	**2158.35**	2158.35	0.06
S2_10i6s	1585.46	16	6	1995.62	17	6	1958.46	0.61	16	**1585.46**	1585.46	0.15	16	5	**1585.46**	0.02	16	5	**1585.46**	1585.46	0.01	16	5	**1585.46**	1585.46	0.04	16	5	**1585.46**	1585.46	0.06
S1_4i2s	1582.20	20	6	1582.20	20	6	**1582.21**	0.63	20	**1582.21**	1582.21	0.13	20	6	1598.91	0.03	20	6	**1582.21**	1582.21	0.01	20	6	**1582.21**	1582.21	0.04	20	6	**1582.21**	1582.21	0.07
S1_4i4s	1460.09	20	6	1580.52	20	5	**1460.09**	0.68	20	**1460.09**	1460.09	0.16	20	5	1483.19	0.03	20	5	**1460.09**	1460.09	0.01	20	5	**1460.09**	1460.09	0.04	20	5	**1460.09**	1460.09	0.07
S1_4i6s	1397.27	20	5	1541.46	20	5	**1397.27**	0.75	20	**1397.27**	1397.27	0.16	20	5	1413.97	0.03	20	5	**1397.27**	1397.27	0.01	20	5	**1397.27**	1397.27	0.04	20	5	**1397.27**	1397.27	0.07
S1_4i8s	1397.27	20	6	1561.29	20	6	**1397.27**	0.82	20	**1397.27**	1397.27	0.17	20	6	**1397.27**	0.03	20	5	**1397.27**	1397.27	0.02	20	5	**1397.27**	1397.27	0.05	20	5	**1397.27**	1397.27	0.07
S1_4i10s	1396.02	20	5	1529.73	20	5	**1396.02**	0.85	20	**1396.02**	1396.02	0.23	20	5	**1396.02**	0.03	20	5	**1396.02**	1396.02	0.02	20	5	**1396.02**	1396.02	0.05	20	5	**1396.02**	1396.02	0.07
S2_4i2s	1059.35	18	5	1117.32	18	4	**1059.35**	0.51	18	**1059.35**	1069.42	0.23	18	4	**1059.35**	0.02	18	4	**1059.35**	1097.64	0.01	18	4	**1059.35**	1059.94	0.04	18	4	**1059.35**	1059.94	0.06
S2_4i4s	1446.08	19	6	1522.72	19	5	**1446.08**	0.60	19	**1446.08**	1449.17	0.21	19	5	**1446.08**	0.02	19	5	1476.91	1476.91	0.02	19	5	**1446.08**	1452.84	0.05	19	5	**1446.08**	1446.08	0.09
S2_4i6s	1434.14	20	6	1730.47	20	5	**1434.14**	0.69	20	**1434.14**	1445.35	0.20	20	5	**1434.14**	0.02	20	5	1440.10	1462.48	0.02	20	5	**1434.14**	1438.43	0.05	20	5	**1434.14**	1435.95	0.08
S2_4i8s	1434.14	20	6	1786.21	20	5	**1434.14**	0.75	20	**1434.14**	1434.14	0.20	20	5	**1434.14**	0.02	20	5	1440.10	1462.48	0.02	20	5	**1434.14**	1439.03	0.05	20	5	**1434.14**	1435.95	0.08
S2_4i10s	1434.13	20	6	1729.51	20	5	**1434.13**	0.78	20	**1434.13**	1455.31	0.24	20	5	**1434.13**	0.02	20	5	1440.09	1462.48	0.02	20	5	**1434.13**	1435.94	0.06	20	5	**1434.13**	1435.94	0.09
Avg. gap above BKS (%)				8.72			0.63			0.00	0.15				0.46				0.09	0.38				0.00	0.04				0.00	0.01	
NBKS				2			38			40					29				35					40					40		
Cum. number of veh.			245			223								225				222					222				222				
Avg. time (min)								0.65				0.17				0.02					0.01					0.04					0.07

Table B.4
Results of MSH on large instances of Erdoğan and Miller-Hooks (2012).

Instance	BKS	MCWS/DBCA			VNS/TS				AVNS				48A				SA			
		n	*v*	Best	*n*	*v*	Best	*t*	*n*	Best	Avg.	*t*	*n*	*v*	Best	*t*	*n*	*v*	Best	*t*
111c_21s	4770.47	109	20	5626.64	109	17	4797.15	21.76	109	**4770.47**	4791.53	1.78	109	18	4960.60	21.74	109	18	5062.06	12.35
111c_22s	4774.65	109	20	5610.57	109	17	4802.16	23.56	109	4776.81	4797.31	1.94	109	18	4914.20	21.23	109	18	5029.17	12.30
111c_24s	4767.14	109	20	5412.48	109	17	4786.96	21.90	109	**4767.14**	4790.84	2.16	109	18	4952.90	21.27	109	18	5010.59	12.13
111c_26s	4767.14	109	20	5408.38	109	17	4778.62	25.12	109	**4767.14**	4782.60	2.04	109	18	4934.11	21.25	109	18	5092.06	12.07
111c_28s	4765.52	109	20	5331.93	109	17	4799.15	24.17	109	**4765.52**	4781.26	1.73	109	18	4971.93	21.29	109	18	5038.84	11.99
200c_21s	8839.62	190	35	10413.59	192	35	8963.46	76.65	192	8886.00	8970.14	3.61	192	32	9276.63	76.41	192	32	9206.28	73.84
250c_21s	10482.52	235	41	11886.61	237	39	10800.18	120.90	237	10487.15	10531.20	3.67	237	39	11007.98	120.67	237	38	10885.71	108.10
300c_21s	12367.60	281	49	14229.92	283	46	12594.77	182.23	283	12374.49	12514.78	4.94	283	46	12869.17	184.27	283	45	12827.35	302.04
350c_21s	14073.34	329	57	16460.30	329	51	14323.02	232.03	329	14103.66	14271.56	7.11	329	54	14954.83	227.30	329	52	14828.63	332.38
400c_21s	16660.20	378	67	19099.04	378	61	16850.21	305.12	378	16697.21	16839.23	12.70	378	61	17351.92	302.03	378	60	17327.27	384.68
450c_21s	18241.48	424	75	21854.17	424	68	18521.23	525.52	424	18310.60	18512.47	13.19	424	68	19215.38	514.68	424	67	19085.91	456.26
500c_21s	20496.50	471	84	24517.08	471	76	21170.90	356.01	471	20609.67	20874.50	19.51	471	76	21636.59	352.16	471	75	21475.71	154.45
Avg. gap above BKS (%)				15.97			1.38			0.17	0.92				4.50				4.97	
NBKS				0			0			4					0				0	
Cum. number of veh.			508			461								466				459		
Avg. time (min)								159.58				6.20				157.03				156.05

Instance	MSH(1k)					MSH(5k)					MSH(10k)				
	n	*v*	Best	Avg.	*t*	*n*	*v*	Best	Avg.	*t*	*n*	*v*	Best	Avg.	*t*
111c_21s	109	17	4798.52	4861.35	0.91	109	17	4780.01	4793.07	2.40	109	17	4777.91	4781.85	4.94
111c_22s	109	17	4795.26	4858.09	1.04	109	17	4776.75	4789.81	2.13	109	17	**4774.65**	4778.80	4.69
111c_24s	109	17	4795.19	4855.89	1.16	109	17	4776.68	4789.77	2.74	109	17	4773.67	4778.62	5.64
111c_26s	109	17	4795.19	4855.89	0.79	109	17	4776.68	4789.77	2.56	109	17	4773.67	4778.62	5.23
111c_28s	109	17	4793.57	4854.86	0.93	109	17	4775.06	4788.22	2.80	109	17	4772.46	4777.03	5.54
200c_21s	192	32	9005.58	9039.97	2.96	192	31	8894.56	8923.18	19.80	192	31	**8839.62**	8879.98	19.96
250c_21s	237	39	10702.76	10755.30	13.43	237	38	10534.52	10579.12	27.59	237	37	**10482.52**	10518.32	21.58
300c_21s	283	45	12663.49	12719.25	44.28	283	44	12444.48	12548.70	37.43	283	44	**12367.60**	12421.75	47.53
350c_21s	329	52	14431.27	14470.40	53.01	329	50	14146.67	14253.55	50.96	329	50	**14073.34**	14226.03	63.01
400c_21s	378	60	16873.51	17143.75	64.29	378	59	16745.24	17128.76	67.91	378	59	**16660.20**	17119.89	71.70
450c_21s	424	66	18569.58	18831.00	72.11	424	65	18351.72	19009.22	76.35	424	65	**18241.48**	18902.03	80.75
500c_21s	471	75	20960.18	21678.33	80.11	471	73	20610.36	21297.07	84.95	471	73	**20496.50**	20997.04	89.95
Avg. gap above BKS (%)			1.42	2.64				0.40	1.48				0.05	1.02	
NBKS			0					0					8		
Cum. number of veh.		454					445					444			
Avg. time (min)					27.92					31.47					35.04

A.2. Notation for multi-space sampling heuristic

G^*	Acyclic graph for the split procedure $G^* = (V^*, A)$
V^*	Set of vertices of G^*
A	Set of arcs of G^*
$\mathcal{K}$	Set of TSP-like tours
$\mathcal{R}$	Set of all feasible routes
Ω	Subset of $\mathcal{R}$
$\mathcal{S}$	Set of all feasible solutions to Green VRP
$\mathcal{H}$	Set of sampling heuristics
K	Number of iterations in sampling phase
tsp	TSP tour ($tsp \in \mathcal{K}$)
s	Green VRP solution ($s \in \mathcal{S}$)
$\mathcal{W}$	Set of customers visited by tsp
$\mathcal{N}$	Ordered set of nonrouted customers
L	Randomization factor of each heuristic
$f(s)$	Objective function of solution s
s^*	Best solution found during execution of method
λ	Positive parameter
c_r	Total distance of route r
tp_r	Total time of route r
f_r	Binary variable equal to 1 if route is feasible
$label_i$	Shortest path label i
$pred_i$	Predecessor label i

A.3. Notation for repair procedure

B	Repair graph $B = (Z, U)$
Z	Set of vertices of B
U	Set of arcs of B
Π	Fuel-infeasible route
$[i, k]$	Vertex of set Z, which represents visit to station $k \in F$ after visit to vertex π_i (i.e., ith element in route Π)
$\bar{c}$	Arc cost of repair graph
$\bar{t}$	Arc time of repair graph

Appendix B. Detailed results for Green VRP instances

Tables B.3 and B.4 show the results of our three MSH configurations on the small and large Green VRP instances. We compare our results to the best result obtained by the MCWS and DBCA heuristics of Erdoğan and Miller-Hooks (2012), the VNS/TS of Schneider et al. (2014), the AVNS of Schneider et al. (2015), and the 48A and SA of Felipe et al. (2014); they did not report result of the SA for the small instances. For each instance, we report the problem name and the BKS taken from Erdoğan and Miller-Hooks (2012), Schneider et al. (2014, 2015), Felipe et al. (2014) and updated with some new BKSs found by our MSH.

Erdoğan and Miller-Hooks (2012) reported the best distance (Best), and the number of vehicles of the best found solution (v) of multiple runs with different parameters for their MCWS and DBCA; they did not give the exact number of runs. Schneider et al. (2014) reported the best distance, the number of vehicles of the best found solution, and the average computing time (t in minutes) over ten runs of their VNS/TS, Schneider et al. (2015) reported the best distance, the average distance (Avg.), and the average computing time over ten runs of their AVNS. Finally, Felipe et al. (2014) reported the best distance, the number of vehicles of the best found solution, and the computing time over a single run of their 48A and SA. For all the algorithms, we provide the number of customers served (n). The last rows of the table summarize the average BKS gap, the cumulative number of vehicles, the number of times each method found the BKS, and the average running time. Values in bold indicate that a method found the BKS.

References

Beasley, J., 1983. Route-first cluster-second methods for vehicle routing. Omega 11 (4), 403–408.
Bräysy, O., Gendreau, M., 2005. Vehicle routing problem with time windows, Part II: Metaheuristics. Transport. Sci. 39 (1), 119–139.
Bullard, J., Kiernan, A., 1992. Plane and Spherical Trigonometry: With Stereographic Projections. D.C. Heath.
Conrad, R.G., Figliozzi, M., 2011. The recharging vehicle routing problem. In: Doolen, T., Aken, E.V. (Eds.), Proceedings of the 2011 Industrial Engineering Research Conference, Reno, NV, USA.
Crevier, B., Cordeau, J.-F., Laporte, G., 2007. The multi-depot vehicle routing problem with inter-depot routes. Eur. J. Oper. Res. 176 (2), 756–773.

Erdoğan, S., Miller-Hooks, E., 2012. A green vehicle routing problem. Transport. Res. Part E: Logist. Transport. Rev. 48 (1), 100–114.
Felipe, Á., Ortuño, M.T., Righini, G., Tirado, G., 2014. A heuristic approach for the green vehicle routing problem with multiple technologies and partial recharges. Transport. Res. Part E: Logist. Transport. Rev., 71: 111–71: 128
Fontecha, J.E., Duque, D., Akhavan-Tabatabaei, R., Rodríguez, J.P., Medaglia, A.L., 2015. Combining maintenance and routing models: the case of a water & sewage utility. Optimization Workshop 2015 (NOW 2015). La Rochelle, France. May 2015.
Foster, B.A., Ryan, D.M., 1976. An integer programming approach to the vehicle scheduling problem. Oper. Res. Quart. 27 (2), 367–384.
Gómez, A., Mariño, R., Akhavan-Tabatabaei, R., Medaglia, A., Mendoza, J.E., 2015. On modeling stochastic travel and service times in vehicle routing. Transport. Sci. http://dx.doi.org/10.1287/trsc.2015.0601.
Gonçalves, F., Cardoso, S.R., Relvas, S., Barbosa-Póvoa, A.P., 2011. Optimization of a distribution network using electric vehicles: A VRP problem. In: IO 2011 – 15° Congresso Nacional da Associação Portuguesa de Investigação Operacional, Lisbon, Portugal.
Juan, A., Goentzel, J., Bektaş, T., 2014. Routing fleets with multiple driving ranges: is it possible to use greener fleet configurations? Appl. Soft Comput., 21:84–21:94
Kleindorfer, P.R., Neboian, A., Roset, A., Spinler, S., 2012. Fleet renewal with electric vehicles at La Poste. Interfaces 42 (5), 465–477.
Laporte, G., Nobert, Y., Desrochers, M., 1985. Optimal routing under capacity and distance restrictions. Oper. Res. 33 (5), 1050–1073.
Lenstra, J.K., Rinnooy Kan, A.H.G., 1981. Complexity of vehicle routing and scheduling problems. Networks 11 (2), 221–227.
Lozano, L., Duque, D., Medaglia, A.L., 2013. The pulse algorithm: a java implementation (jpulse). <http://dspace.uniandes.edu.co:9090/xmlui/handle/1992/1162> (accessed 05/14/2014).
Lozano, L., Medaglia, A.L., 2013. On an exact method for the constrained shortest path problem. Comput. Oper. Res. 40 (1), 378–384.
Mendoza, J.E., Villegas, J.G., 2013. A multi-space sampling heuristic for the vehicle routing problem with stochastic demands. Optim. Lett. 7 (7), 1503–1516.
Muter, I., Cordeau, J.-F., Laporte, G., 2014. A branch-and-price algorithm for the multidepot vehicle routing problem with interdepot routes. Transport. Sci. 48 (3), 425–441.
Nesterova, N., Quak, H., Balm, S., Roche-Cerasi, I.T., Tretvik, T., 2013. State of the art of the electric freight vehicles implementation in city logistics. Technical Report FREVUE D1.3, TNO.
Pelletier, S., Jabali, O., Laporte, G., 2014. Goods distribution with electric vehicles: review and research perspectives. Technical Report CIRRELT-2014-44, CIRRELT, Montréal, Canada.
Pollet, B.G., Staffell, I., Shang, J.L., 2012. Current status of hybrid, battery and fuel cell electric vehicles: from electrochemistry to market prospects. Electrochim. Acta, 84:235–84:249.
Prins, C., 2004. A simple and effective evolutionary algorithm for the vehicle routing problem. Comput. Oper. Res. 31 (12), 1985–2002.
Prins, C., Lacomme, P., Prodhon, C., 2014. Order-first split-second methods for vehicle routing problems: a review. Transport. Res. Part C: Emerg. Technol. 40, 179–200.
Priselac, M., 2013. A frigid fleet: Coca-cola unveils first electric refrigerated trucks. <http://www.coca-colacompany.com/stories/a-frigid-fleet-coca-cola-unveils-first-electric-refrigerated-trucks> (accessed 05/14/2014).
Sassi, O., Cherif, W.R., Oulamara, 2014. Vehicle routing problem with mixed fleet of conventional and heterogenous electric vehicles and time dependent charging costs. Technical Report, hal-01083966.
Schneider, M., Stenger, A., Goeke, D., 2014. The electric vehicle-routing problem with time windows and recharging stations. Transport. Sci. 48 (4), 500–520.
Schneider, M., Stenger, A., Hof, J., 2015. An adaptive VNS algorithm for vehicle routing problems with intermediate stops. OR Spectr. 37 (2), 353–387.
Tarantilis, C., Zachariadis, E., Kiranoudis, C., 2008. A hybrid guided local search for the vehicle-routing problem with intermediate replenishment facilities. INFORMS J. Comput. 20 (1), 154–168.
Thompson, P.M., Psaraftis, H.N., 1993. Cyclic transfer algorithm for multivehicle routing and scheduling problems. Oper. Res. 41 (5), 935–946.
UPS, 2008. UPS hybrid electric vehicle fleet. <http://www.pressroom.ups.com/HEV> (accessed 05/14/2014).
Vidal, T., Maculan, N., Ochi, L., Penna, P.H.V., 2015. Large neighborhoods with implicit customer selection for vehicle routing problems with profits. Transport. Sci. http://dx.doi.org/10.1287/trsc.2015.0584.
Villegas, J.G., Prins, C., Prodhon, C., Medaglia, A.L., Velasco, N., 2010. GRASP/VND and multi start evolutionary local search for the single truck and trailer routing problem with satellite depots. Eng. Appl. Artif. Intell. 23 (5), 780–794.