Contents lists available at GrowingScience

International Journal of Industrial Engineering Computations

homepage: www.GrowingScience.com/ijiec

Variable neighborhood search algorithm for the green vehicle routing problem

Mannoubia Affi[a], Houda Derbel[a*] and Bassem Jarboui[b]

[a]*MODILS, FSEGS, Route de l'aéroport km 4, Sfax 3018 Tunisia*
[b]*Emirates College of Technology, Abu Dhabi, United Arab Emirates*

C H R O N I C L E

Article history:
Received February 13 2017
Received in Revised Format April 1 2017
Accepted June 16 2017
Available online June 16 2017

Keywords:
Green vehicle routing problem
Refueling stations
Variable neighborhood search
Heuristics

A B S T R A C T

This article discusses the ecological vehicle routing problem with a stop at a refueling station titled Green-Vehicle Routing Problem. In this problem, the refueling stations and the limit of fuel tank capacity are considered for the construction of a tour. We propose a variable neighborhood search to solve the problem. We tested and compared the performance of our algorithm intensively on datasets existing in the literature.

1. Introduction

Transportation is one of the most important aspects of logistics and basic infrastructure for the economic growth. However, it was considered one among the huge consumers of petroleum and represents a large portion of overall pollutants. Many logistic companies have already started to establish green logistic projects to reduce CO2 emissions. The green vehicle routing problem (G-VRP) is characterized by examining the environmental areas and the costs estimated to implement effective routes to respond to environmental and financial concerns. Travel costs are given by two physical quantities considered to be representative of the ecological imprint: reducing greenhouse gas emissions (CO2) and energy savings (fossil energy, renewable energy...). The G-VRP deals with models aiming at minimizing fuel consumption. It was introduced by Erdogan et al. (2012) who examined the possibility of recharging a vehicle fuel within a fixed refueling time and without capacity constraints or time window restrictions. The authors presented two solution heuristics to solve the problem namely a modified Clarke and Wright savings algorithm (MCWS) that handles infeasible routes by inserting alternative fuel stations (AFS) using a savings criterion and removing redundant AFSs by merging the routes and a density-based clustering algorithm (DBCA) which is a cluster-first and route-second approach.

* Corresponding author Tel.: +216-98-972101
E-mail: derbelhouda@yahoo.fr (H. Derbel)

doi: 10.5267/j.ijiec.2017.6.004

The vehicle routing problem with intermediate stops (VRPIS) was introduced by Schneider et al. (2015) where intermediate facilities are visited depending on the fuel and/or the load level of a delivery vehicle. The authors developed an adaptive variable neighborhood search (AVNS) to deal with this problem. Two problems were considered as special cases of VRPIS namely the GVRP and the electric VRP with recharging facilities (EVRPRF). Schneider et al. (2014) proposed a variable neighborhood search (VNS) with a tabu search (TS) to solve the electric vehicle routing problem with time windows and recharging stations (E-VRPTW). The VNS/TS algorithm was able to improve the results of Erdogan et al. (2012).

Bruglieria et al. (2015) proposed a mixed formulation of linear integer programming of electric vehicle routing problem with time windows, which assumes that the level of recharging the battery of the vehicle at each station is a decision variable to ensure more flexible tours. The objective function was to minimize the total distance, waiting times and the number of electric vehicles.

A multi-space sampling heuristic (MSH) for the GVRP was developed recently by Montoya et al. (2016). The algorithm alternates between two phases: sampling and assembling. During the sampling phase, sampled TSP tours are built first and feasible solutions are extracted second by solving a set partitioning formulation. The assembling procedure is used a set partitioning model to assemble sampled routes in the sampling phase. Experimental results show that MSH reported 8 new best known solutions (BKSs) among 52 ones and give competitive results with respect to MCWS/ DBCA, VNS/TS, AVNS.

In addition to the studies mentioned above, several models for fuel consumption and emissions of greenhouse gases in the road freight transport have been analyzed in the work of Demir et al. (2011). More specifically, the authors compared six models and evaluated their respective strengths and weaknesses. These models indicate that fuel consumption depends on a number of factors that can be grouped into four categories: vehicle, driver, environment and traffic. Among the extensions of the G-VRP which aims to reduce CO2 emissions, we cite for example the pollution routing problem (PRP) which is based on the global model of emissions with less pollution, especially reducing CO2 emissions. This has been proposed by Bektas et al. (2011). They developed a global objective function that includes the minimization of the cost of CO2 emissions and operating costs for drivers and fuel consumption. In estimating pollution, factors such as speed, load and time windows are considered. Several extensions that extend the PRP problem have been introduced after (Franceschetti et al., 2013; Demir et al., 2014).

In our work, we study the G-VRP modeled to help organizations with alternative fuel-powered vehicle fleets (AFV) to overcome the difficulties that exist as a result of the limited refueling infrastructure. Consequently, the objective is to minimize the total distance to serve a set of customers by integrating AFS nodes at each route in order to eliminate the risk of running out of fuel. We develop a variable neighborhood search (VNS) which combines a successful local search, namely a sequential variable neighborhood descent (Seq-VND) with effective neighborhood structures used in the shaking process. The algorithm is implemented and tested on instances of GVRP studied in the literature.

The remainder of this paper is organized as follows: In section 2, we detail the definition of the problem. Section 3 presents the different steps within the VNS algorithm. A presentation of our computational results is provided in Section 4. Finally, we conclude the paper with section 5.

2. Problem definition

Green-VRP is defined by an undirected graph G = (V, E), where the set of nodes V consists of the set of customers $I = \{v_1, v_2, ..., v_n\}$, the depot v_0, a set of AFS nodes $F = \{v_{n+1}, v_{n+2}, ..., v_{n+s}\}$ where $s \geq 0$ and a set of fictitious nodes $\Phi = \{v_{n+s+1}, v_{n+s+2}, ..., v_{n+s+s'}\}$, $s' \geq 0$, one for each potential visit of a station or depot. Each refueling station $v_f \in F$ is associated with a set of fictitious nodes n_f, with $f \in \{0, ..., n+s\}$. The number n_f corresponds to the number of times the node v_f can be visited. The set of nodes $V = \{v_0\} \cup I \cup F \cup \Phi = \{v_0, v_1, v_2, ..., v_{n+s+s'}\}$, $|V| = n + s + s' + 1$. It is assumed that in addition to alternative fuel stations, the depot can be used as a refueling station and all refueling stations have unlimited capacity. The set E = $\{(v_i, v_j) : v_i, v_j \in V, i < j\}$ corresponds to edges connecting nodes of V. Each edge (v_i, v_j) is associated with a cost c_{ij}, distance d_{ij} and a travel time t_{ij}. The problem involves designing a set of vehicle routes, each one starting and ending at the depot with visiting a subset of nodes containing AFSs when necessary such that the total distance traveled is minimized, the depot must be visited at the beginning and the end of each route and can be used as a refueling station if desired. Moreover, each AFS can be visited more than once or not at all. Furthermore, each customer should be visited exactly once. The travel speeds are assumed to be constant over a link. It is assumed that the tank is filled to capacity when refueling is performed. The main constraints corresponding to the GVRP are as follows:

- a tour is built such that each customer node has exactly one successor: a customer, a station or a depot and each station as well as the fictitious node associated to it have at most one successor node: a customer, a station or a depot,
- at most m vehicles are routed out of the depot and at most m vehicles return to the depot in a given day,
- each tour is completed by a maximal time $T_{\max}$,
- the fuel level is limited to Q when vehicles arrive to depot or at refueling station.

The reader can see (Erdogan et al., 2012) for more details about the mathematical formulation and different constraints of the G-VRP.

3. Variable neighborhood Search for the G-VRP

Mladenovic et al. (1997) introduced a simple but powerful metaheuristic called Variable Neighborhood Search (VNS) based on the principle of systematic change of neighborhood. Since then, several variants of VNS were developed to solve combinatorial optimization problems such as location-routing problem (Fagerholt et al., 2010). Two main features characterize the VNS algorithm specifically a perturbation (or shaking) phase (diversification) and a local search phase (intensification). The VNS algorithm was conceived to explore the solution space by successively applying the two phases to the current solution (Hansen et al., 2010).

In this paper, we propose a general VNS (GVNS) (see algorithm 1 based on a variable neighborhood descent (VND)) as a local search phase. We observe that the tours tend to be complex and intertwined because the refueling of vehicles and customers visits should be programmed to respect the maximum length of tours and the level of fuel remaining in the tank. To overcome these constraints, we define a set of different neighborhood structures within the solution space according to the different moves of customers and refueling stations.

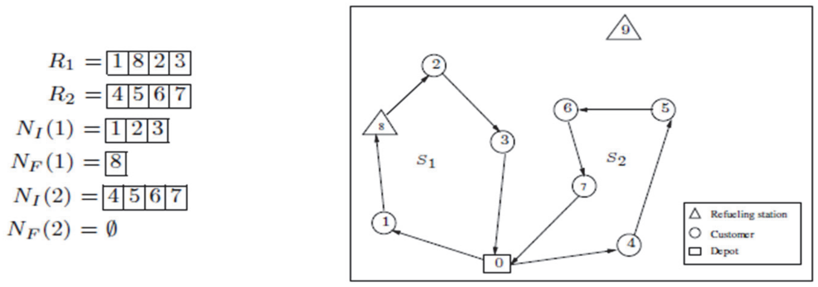
Algorithm 1: GVNS general structure

```
1   Initialization. Generate an initial solution $S$
2   Select the set of neighborhood structures $N_k$,$N_l$ , k = 1, ..., $k_{\max}$, l = 1, ..., $l_{\max}$
3   Choose a stopping criterion;
4   while Termination condition is met do
5       k ← 1
6       repeat
7           Shaking
8           Generate a solution $S' \in N_k(S)$ randomly ;
9           Local Search with VND
10          l ← 1;
11          repeat
12              Find the best neighbor $S'' \in N_l(S')$;
13              if f($S''$) < f($S'$) then $S' \leftarrow S''$ and l = 1 else l ← l + 1;
14          until l = $l_{\max}$ ;
15          Evaluation
16          if f($S''$) < f($S$) then $S \leftarrow S''$ and k = 1 else k ← k + 1;
17      until k = $k_{\max}$ ;
```

3.1 Solution representation and Neighborhood structures

We use the following notations to better explain one solution of the GVRP and the neighborhood structures used in our algorithm. Given a solution S, let m be the number of the used vehicles, I be the number of the visited customers, F be the number of the visited refueling stations and $R = I \cup F$ be the total number of visited nodes in the solution. More precisely, a solution $S = (S_1, ..., S_m)$ of our problem is presented by a sequence of m routes where each route $S_v = (S_{v1}, ..., S_{vN(v)})$ represents the set of N (v) nodes visited by the vehicle v. Each element S_{vk} corresponds to the kth node visited by vehicle v. N $(v) = (N_I(v) \cup N_F(v))$ where $N_I(v)$, $N_F(v)$ are the sets of customers and refueling stations visited by the vehicle v respectively.

Consider the case of a G-VRP with two refueling stations and seven customers. Fig. 1 illustrates an example of a G-VRP solution. The refueling station 8 is visited while visiting the customers1, 2 and 3 in the route S_1 whereas the refueling station 9 is not visited.



$R_1 = \boxed{1}\boxed{8}\boxed{2}\boxed{3}$
$R_2 = \boxed{4}\boxed{5}\boxed{6}\boxed{7}$
$N_I(1) = \boxed{1}\boxed{2}\boxed{3}$
$N_F(1) = \boxed{8}$
$N_I(2) = \boxed{4}\boxed{5}\boxed{6}\boxed{7}$
$N_F(2) = \emptyset$
9
2
8
3
6
5
S_1
S_2
7
1
0
4
Refueling station
Customer
Depot

Fig. 1. An example of G-VRP solution representation

Specifically, a neighborhood is built by modifying certain elements of a given solution to create a new neighboring solution. However, a local optimal solution in a neighborhood structure is not necessary a local optimum in another neighborhood structure. For this reason, the use of several adjacent structures help to guide the local search to converge to a local optimum. Define the following three moves to describe each neighborhood used in our algorithm: 1. Drop(S_v, k): consists of removing the node at position k in the route S_v, 2. Insert(S_v, k, i): consisting of inserting the node i at position k in the route S_v, 3. Exchange(S_v, k, i) consisting of changing the node at position k by a new node i in the route S_v.

In the sequel, we briefly describe the different neighborhood structures we have used in our VNS algorithm.

3.1.1 Customer neighborhoods

a. Shift move of a customer neighborhood (N_1): A neighbor of a solution S is obtained by removing a customer from its position and inserting it in a different position, within the same route or in another route. It corresponds to a shift move of a customer.

$N_1(S) = \{S', \forall v, v' \in$ M, k $\in N_I(v)$, $k' \in N_I(v')$, S'_v = Drop(S_v, k), $S'_{v'}$ = Insert($S_{v'}$, k', S_{vk})$\}$

b. Swap move of two customers neighborhood (N_2): A neighbor of a solution S is obtained by exchanging two customers, from the same route or from different routes. It corresponds to a swap move of two customers.

$N_2(S) = \{S', \forall v, v' \in$ M, k $\in N_I(v)$, $k' \in N_I(v')$, S'_v = Exchange(S_v, k, $S_{v'k'}$), $S'_{v'}$ = Exchange($S_{v'}$, k', S_{vk})$\}$

3.1.2 Refueling station neighborhoods

a. Insertion neighborhood (N_3): A neighbor of solution S is obtained by inserting any refueling station in a route.

$N_3\ (S) = \{S', \forall v \in$ M, k ∈ N (v), i ∈ F, S'_v = Insert(S_v , k, i)}

b. Drop neighborhood (N_4): A neighbor of a solution S is obtained by removing a refueling station of a route.

$N_4\ (S) = \{S'$, $\forall v \in$ M, k ∈ N (v), S'_v = Drop(S_v, k)}

c. Replacement neighborhood (N_5): A neighbor of a solution S is obtained by replacing a refueling station of one route by another refueling station.

$N_5\ (S) = \{S'$, $\forall v \in$ M, k ∈ N_{F} (v), i ∈ F, S'_v= Insert(Drop(S_v , k), k, i)}

d. Shift neighborhood (N_6):A neighbor of a solution S is obtained by removing a refueling station from its route and shifting it to a new position in the same route.

$N_6\ (S) = \{S'$, $\forall v \in$ M, k ∈ N (v), $k' \in$ N $(v'$), i ∈ R, S'_v = Drop(S_v , k), $S'_{v'}$ = Insert(S'_v , k' , S_{vk})}

3.1.3 Node neighborhoods

a. Shift node neighborhood (N_7): A neighbor of a solution S is obtained by choosing two nodes randomly from the same route or from two different routes, then removing one of those two nodes from its position and inserting it in the position before the position of the other node.

$N_7\ (S) = \{S'$, $\forall v, v' \in$ M, k ∈ N_{F} (v), $k' \in$ N $(v'$), i ∈ F, S'_v = Drop(S_v , k), $S'_{v'}$ = Insert(S'_v , k' , S_{vk})}

b. Random crossing neighborhood (N_8): A neighbor of a solution S is obtained by considering two nodes i and j selected randomly in two different routes called crosspoints. We obtain a new solution by removing the arc (i, j) and recombining the remaining parts. The steps of N_8 are summarized in algorithm 2

Algorithm 2: Steps of the neighborhood N_8

```
1   Input. $S'$ , $\forall v, v' \in$ M, k ∈ N $(v)$, $k' \in$ N $(v')$
2   i ← k + 1;
3   j ← $k'$ ;
4   while i < N $(v)$ and j < N $(v')$ do
5       $S'_v$ = Exchange($S_v$ , i, $S_{v'j}$ );
6       $S'_{v'}$ = Exchange($S_{v'}$ , j, $S_{vi}$ );
7       i ← i + 1;
8       j ← j + 1;
```

c. Crossing neighborhood (N_9): A neighbor of a solution S is obtained by taking two nodes having the minimum Euclidean distance belonging to two different routes chosen at random, called breakpoints, then combining the two parts one that ends with the first cut-off point with the other starting from the second cut-off point and the remaining two parts together.

3.2 Evaluation function

We allow our VNS algorithm to visit infeasible solutions that exceed the limit time for each route or that exceed the fuel tank capacity. Given a solution S, let *m* be the number of routes and N (v), $v \in \{1, ..., \mathrm{m}\}$ be the number of visited nodes in a given route v in the solution S. Let f (S) be the evaluation function of a solution S, f (S) is defined as follows: f (S) = C (S) + p_1 (S) + p_2 (S) such that C (S) represents the total distance of the G-VRP solution, p_1 (S) is a penalty on the violation of the limit time constraints and p_2 (S) represents a penalty on the violation of the fuel tank capacity constraints. More precisely, p_1 (S) and p_2 (S) are formulated as follows:

$$p_1\,(S)=\alpha \max\left\{0,\ \sum_{v=1}^{M}\left(\sum_{k=1}^{N(v)} T_{vk}(S) - T_{max}\right)\right\}$$

$$p_2\,(S)=\beta \max\left\{0,\ \sum_{v=1}^{M}\sum_{k\in N_F(v)\cup\{0\}}(Q_{vk}(S) - Q)\right\}$$

where $T_{vk}(S)$ and $Q_{vk}(S)$ are the total time and the total used fuel at node k of a given route v in the solution S, respectively. T_{max} is the duration limit of each route, Q is the fuel tank capacity of the vehicle and the parameters α and β are constant factors that present the degree of considered penalties.

3.3 Local search phase

In our work, we implement a local search corresponding to a variable neighborhood descent algorithm (VND) algorithm (Hansen et al. 2006). VND is the multi-neighborhood version of the simple local search, where the neighborhoods N_i,i∈{1, ..., 6} are used sequentially to improve the current solution. More precisely, we start the VND algorithm by generating an initial solution S at random, this solution will be improved sequentially by applying the first neighborhood N_1 , and so forth we proceed in the same way with the other neighborhoods (see Algorithm 3) . The VND algorithm stops when no more improvement is possible.

Algorithm 3: Sequential-VND

```
1   Input. Set of neighborhood structures $N_k$,k∈{1...6} , a random initial solution $S$
2   k ← 1;
3   while k ≤ 6 do
4       $S'$ ← the first improvement on $S$ using neighborhood $N_k$ ;
5       if f($S'$) < f($S$) then $S$ ← $S'$ and k ← 1;
6       else k ← k + 1;
```

3.4 Shaking phase

After a local search phase, a shaking phase will be performed. Neighborhoods N_k,k∈{7,8,9} are used to perturb a local optimum reached in the local search. Indeed, we perturb the current solution at each iteration by choosing randomly one of the three neighborhood structures according to a probability Pr(N_k), i ∈ {7, 8, 9}. The steps of the shaking phase are summarized in the algorithm 4.

Algorithm 4: Shaking

```
1   Input. Set of neighborhood structures $N_k$,k∈{7,8,9} , P r($N_k$ ),
2   S: local optimum reached by Sequential VND
3   for k ≤ $k_{max}$ do
4       P r : a randomly selected probability;
5       if P r > P r($N_7$ ) then Generate $S' \in N_7(S)$;
6       Else
7       if P r($N_8$) < P r ≤ P r($N_7$ ) then Generate $S' \in N_8$ (S);
8       Else
9       Generate $S' \in N_9$(S);
10      if f($S'$) < f($S$) then S ← $S'$
```

3.5 VNS algorithm

Our VNS approach mainly involves the three steps of the GVNS as already described in algorithm 1: initialization, shaking, VND local search. More precisely, we begin the algorithm by initializing a solution S randomly and the set of neighborhoods N_k, k ∈ {1, ..., 9} . Then a solution S' is obtained by applying the VND described in section 3.3. After that, we apply the shaking phase as described in section 3.4 to reach a solution S''. If f (S'') < f (S), the solution S is updated and the shaking is repeated, k = 1. If there is no improvement, k = k + 1, return to the VND, and the process continues. The whole procedure of the VNS algorithm is repeated until the stopping condition is met. In our VNS, we use a maximal time, $CPU_{\max}$ as a stopping condition. The pseudocode of our VNS algorithm for solving the GVRP is given in Algorithm 5 below.

Algorithm 5: VNS pseudocode

```
1 Input. Set of neighborhood structures $N_k$,k∈{1,...,9}
2 Initialization. Find an initial solution S randomly
3 Repeat
4     k ← 1
5     while k ≤ $k_{\max}$ do
6         $S'$ ← Sequential-VND($S$);
7         $S''$ ← Shaking($S'$);
8         if f ($S''$) < f (S) then S ← $S''$ and k = 1 else k ← k + 1;
9 until C P U < $CPU_{\max}$ ;
```

4. Computational results

Our algorithm was implemented in C ++ and runs with an Intel (R) Core (TM) i5-4460 CPU, 3.20GHz. We test our algorithm on two sets of instances mentioned in the literature (Erdogan et al., 2012). The first set includes 40 instances of small size with 20 customers and the second one contains 12 instances with up to 500 customers. For the shaking phase, the probability of choosing one of the neighborhoods N_7 , N_8 and N_9 are Pr(N_7) = 0.75, Pr(N_8) = 0.50 and Pr(N_9) = 0, 25 respectively. The VNS algorithm was run 10 times for each of the small instances and a single run for large instances. Its termination condition corresponds to a time limit $CPU_{\max} = 50 * $ n seconds where n is the number of customers. We fix $k_{\max}$ = 2 in the shaking phase by means of a parameter adjustment work. Within the scope of our work, we force our algorithm to visit feasible solutions by fixing large values for the parameters α and β, α = β = 1000.

In this section, a computational study is carried out to compare our approach with best known solutions. According to the computational experiments, our algorithm is competitive with best results mentioned in the literature for small and large instances of G-VRP. Table 1 reports the computational results on the small set of Green-VRP instances and Table 2 reports the best results on large instances of G-VRP. The first column reports the instance name. The best known results in the literature are presented in the column BKS. The remainder of the columns report the results obtained with MCWS/DBCA, VNS/TS, AVNS and MSH. As for the MSH, the authors have conducting results for three different values of iterations in the sampling phase namely: 1000, 5000 and 10000. We compare our results to those given by the best one MSH(10000). We denote by f the best result obtained during the 10 runs, *n* the number of visited customers in the solution, m the number of used vehicles, gap the average deviation relative to the best known solution and t the running time in minutes. Avg.gap (%) and Avg.time (%) represent the average gap above BKS and the average computational time in minutes respectively. The improved values of f are underlined.When compared with the results provided in the literature, VNS provides the best solution value for all instances within less time for small instances. It is also mentioned that, we are able to reduce the number of used vehicles for one instance.

Table 1
Results on small instances of G-VRP

Instance	BKS	MCWS/DBCA			VNS/TS				A VNS				MSH					Ours				
		Bes	*n*	*m*	*Best*	*n*	*m*	*t*	*Best*	*n*	*Avg*	*t*	*Best*	*n*	*m*	*Avg*	*t*	*Best*	*n*	*m*	*Avg*	*t*
20*c*3*sU* 1	1797,49	1797.51	20	6	1797.49	20	6	0.69	1797.49	20	1797.49	0.16	1797.49	20	6	1797.49	0.08	**1797,49**	20	6	**1797,49**	0,0014
20*c*3*sU* 2	1574.77	1613.53	20	6	1574.77	20	6	0.64	1574.78	20	1574.78	0.15	1574.78	20	6	1574.78	0.07	**1574,77**	20	6	**1574,77**	0,0004
20*c*3*sU* 3	1704.48	1964,57	20	7	1704.48	20	6	0.64	1704.48	20	1704.48	0.13	1704.48	20	6	1704.48	0.07	**1704,48**	20	6	**1704,48**	0,0026
20*c*3*sU* 4	1482.00	1487.15	20	6	1482.00	20	5	0.65	1482.00	20	1482.00	0.17	1482.00	20	5	1482.00	0.07	**1482,00**	20	5	**1482,00**	0,0021
20*c*3*sU* 5	1689.37	1752.73	20	5	1689.37	20	6	0.67	1689.37	20	1689.37	0.18	1689.37	20	6	1689.37	0.07	**1689,37**	20	6	**1689,37**	0,0013
20*c*3*sU* 6	1618.65	1668.16	20	6	1618.65	20	6	0.67	1618.65	20	1618.65	0.15	1618.65	20	6	1618.65	0.07	**1618,65**	20	6	**1618,65**	0,0004
20*c*3*sU* 7	1713.66	1730.45	20	6	1713.66	20	6	0.64	1713.66	20	1713.66	0.19	1713.66	20	6	1713.87	0.07	**1713,66**	20	6	**1713,66**	0,0005
20*c*3*sU* 8	1706.50	1718,67	20	6	1706.50	20	6	0.67	1706.50	20	1706.50	0.16	1706.50	20	6	1706.50	0.07	**1706,50**	20	6	**1706,50**	0,0012
20*c*3*sU* 9	1708.81	1714.43	20	6	1708.81	20	6	0.66	1708.82	20	1708.82	0.19	1708.82	20	6	1709.65	0.07	**1708,81**	20	6	**1708,81**	0,0008
20*c*3*sU* 10	1181.31	1309.52	20	5	1181.31	20	4	0.64	1181.31	20	1181.31	0.23	1181.31	20	4	1181.31	0.07	**1181,31**	20	4	**1181,31**	0,0006
20*c*3*sC*1	1173.57	1300.62	20	5	1173.57	20	4	0.62	1173.57	20	1173.57	0.38	1173.57	20	4	1173.57	0.07	**1173,57**	20	40	**1173,57**	0,0006
20*c*3*sC*2	1539.97	1553.53	19	5	1539.97	19	5	0.58	1539.97	19	1539.97	0.21	1539.97	19	5	1539.97	0.08	**1539,96**	19	5	**1539,96**	0,0014
20*c*3*sC*3	880.20	1083.12	12	4	880.20	12	3	0.25	880.20	12	880.20	0.15	880.20	12	3	880.20	0.04	**880,20**	12	3	**880,20**	0,0006
20*c*3*sC*4	1059.35	1091.78	18	5	1059.35	18	4	0.53	1059.35	18	1077.71	0.23	1059.35	18	4	1059.94	0.06	**1059,35**	18	4	**1059,35**	0,0008
20*c*3*sC*5	2156.01	2190.68	19	7	2156.01	19	7	0.60	2156.01	19	2156.01	0.14	2156.01	19	7	2156.04	0.10	**2156,01**	19	7	**2156,01**	0,0006
20*c*3*sC*6	2758.17	2883.71	17	9	2758.17	17	8	0.71	2758.17	17	2758.17	0.14	2758.17	17	8	2758.17	0.08	**2758,17**	17	8	**2758,17**	0,0021
20*c*3*sC*7	1393.99	1701.40	6	5	1393.99	6	4	0.18	1393.99	6	1393.99	0.04	1393.99	6	4	1393.99	0.06	**1393,99**	6	4	**1393,99**	0,001
20*c*3*sC*8	3139.72	3319.74	18	10	3139.72	18	9	0.62	3139.72	18	3139.72	0.08	3139.72	18	9	3139.72	0.12	**3139,72**	18	9	**3139,72**	0,0006
20*c*3*sC*9	1799.94	1811.05	19	6	1799.94	19	6	0.60	1799.94	19	1799.94	0.16	1799.94	19	6	1799.94	0.10	**1799,94**	19	6	**1799,94**	0,0005
20*c*3*sC*10	2583.42	2644.11	15	8	2583.42	15	8	0.45	2583.42	15	2600.39	0.09	2583.42	15	8	2583.42	0.07	**2583,42**	15	9	**2583,42**	0,0011
*S*1−2*i*6*s*	1578.12	1614.15	20	6	1578.12	20	6	0.71	1578.12	20	1578.12	0.16	1578.12	20	6	1578.12	0.07	**1578,12**	20	6	**1578,12**	0,0013
*S*1−4*i*6*s*	1397.27	1541.46	20	5	1397.27	20	5	0.75	1397.27	20	1397.27	0.16	1397.27	20	5	1397.27	0.07	**1397,27**	20	5	**1397,27**	0,0006
*S*1−6*i*6*s*	1560.49	1616.20	20	6	1560.49	20	5	0.73	1560.49	20	1560.49	0.20	1560.49	20	5	1560.49	0.07	**1560,49**	20	5	**1560,49**	0,0039
*S*1−8*i*6*s*	1692.32	1882.54	20	6	1692.32	20	6	0.74	1692.32	20	1692.32	0.17	1692.32	20	6	1692.32	0.07	**1692,32**	20	6	**1692,32**	0,0017
*S*1−10*i*6*s*	1173.48	1309.52	20	5	1173.48	20	4	0.71	1173.48	20	1173.48	0.24	1173.48	20	4	1173.48	0.07	**1173,48**	20	4	**1173,48**	0,0007
*S*2−2*i*6*s*	1633.10	1645.80	20	6	1633.10	20	6	0.75	1633.10	20	1633.10	0.19	1633.10	20	6	1633.10	0.09	**1633,09**	20	6	**1633,09**	0,0007
*S*2−4*i*6*s*	1505.07	1505.07	19	6	1532.96	19	5	0.88	1505.07	19	1505.07	0.14	1505.07	19	6	1505.07	0.09	**1505,06**	19	5	**1505,06**	0,0002
*S*2−6*i*6*s*	2431.33	3115.10	20	10	2431.33	20	7	0.78	2431.33	20	2431.33	0.13	2431.33	20	7	2431.33	0.07	**2431,33**	20	7	**2431,33**	0,0005
*S*2−8*i*6*s*	2158.35	2722.55	16	9	2158.35	16	7	0.57	2158.35	16	2158.35	0.09	2158.35	16	7	2158.35	0.06	**2158,35**	16	7	**2158,35**	0,0007
*S*2−10*i*6*s*	1585.46	1995.62	16	6	1958.46	17	6	0.61	1585.46	16	1585.46	0.15	1585.46	16	5	1585.46	0.06	**1585.46**	17	6	**1585.46**	0,0022
*S*1−4*i*2*s*	1582.20	1582.20	20	6	1582.21	20	6	0.63	1582.21	20	1582.21	0.13	1582.21	20	6	1582.21	0.07	**1582,20**	20	6	**1582,20**	0,0007
*S*1−4*i*4*s*	1460.09	1580.52	20	6	1460.09	20	5	0.68	1460.09	20	1460.09	0.16	1460.09	20	5	1460.09	0.07	**1460,09**	20	5	**1460,09**	0,0016
*S*1−4*i*6*s*	1397.27	1541.46	20	5	1397.27	20	5	0.75	1397.27	20	1397.27	0.16	1397.27	20	5	1397.27	0.07	**1397,27**	20	5	**1397,27**	0,0006
*S*1−4*i*8*s*	1397.27	1561.29	20	6	1397.27	20	6	0.82	1397.27	20	1397.27	0.17	1397.27	20	5	1397.27	0.07	**1397,27**	20	5	**1397,27**	0,0011
*S*1−4*i*10*s*	1396.02	1529.73	20	5	1396.02	20	5	0.85	1396.02	20	1396.02	0.23	1396.02	20	5	1396.02	0.07	**1396,02**	20	5	**1396,02**	0,001
*S*2−4*i*2*s*	1059.35	1117.32	18	5	1059.35	18	4	0.51	1059.35	18	1069.42	0.23	1059.35	18	4	1059.94	0.06	**1059,35**	18	4	**1059,35**	0,0044
*S*2−4*i*4*s*	1446.08	1522.72	19	6	1446.08	19	5	0.60	1446.08	19	1449.17	0.21	1446.08	19	5	1446.08	0.09	**1446,08**	19	5	**1446,08**	0,0031
*S*2−4*i*6*s*	1434.14	1730.47	20	6	1434.14	20	5	0.69	1434.14	20	1445.35	0.20	1434.14	20	5	1435.95	0.08	**1434,14**	20	5	**1434,14**	0,0021
*S*2−4*i*8*s*	1434.14	1786.21	20	6	1434.14	20	5	0.75	1434.14	20	1434.14	0.20	1434.14	20	5	1435.95	0.08	**1434,14**	20	5	**1434,14**	0,0039
*S*2−4*i*10*s*	1434.13	1729.51	20	6	1434.13	20	5	0.78	1434.13	20	1455.31	0.24	1434.13	20	5	1435.94	0.09	**1434,13**	20	5	**1434,13**	0,0035
Avg.gap		8.72			0.63				0.00		0.15		0.00			0.01		0.00			0.00	
Avg.time								0.65				0.17					0.07					0.0014

Table 2
Results on large instances of G-VRP

Instance	BKS	MCWS/DBCA			VNS/TS				AVNS				MSH					Our best results				
		Best	*n*	*m*	*Best*	*n*	*m*	*t*	*Best*	*n*	*Avg*	*t*	*Best*	*n*	*m*	*Avg*	*t*	*Best*	*n*	*m*	*Avg*	*t*
111*c*−21*s*	4770.47	5626.64	109	20	4797.15	109	17	21.76	4770.47	109	4791.53	1.78	4777.91	109	17	4781.85	4.94	**4770.46**	109	17	4802,46	0.49
111*c*−22*s*	4774.65	5610.57	109	20	4802.16	109	17	23.56	4776.81	109	4797.31	1.94	4774.65	109	17	4778.8	4.69	**4768.93**	109	17	4770,76	0.14
111*c*−24*s*	4767.14	5412.48	109	20	4786.96	109	17	21.9	4767.14	109	4790.84	2.16	4773.67	109	17	4778.62	5.64	**4767.14**	109	17	4767,18	2.84
111*c*−26*s*	4767.14	5408.38	109	20	4778.62	109	17	25.12	4767.14	109	4782.6	2.04	4773.67	109	17	4778.62	5.23	**4767.14**	109	17	4772,54	3.17
111*c*−28*s*	4765.52	5331.93	109	20	4799.15	109	17	24.17	4765.52	109	4781.26	1.73	4772.46	109	17	4777.03	5.54	**4765.52**	109	17	4781,03	4.37
200*c*−21*s*	8839.62	10413.59	190	35	8963.46	192	35	76.65	8886	192	8970.14	3.61	8839.62	192	31	8879.98	19.96	**8789.09**	192	31	8845,06	11.8
250*c*−21*s*	10482.52	11886.61	235	41	10800.18	237	39	120.9	10487.15	237	10531.2	3.67	10482.52	237	37	10518.32	21.58	**10485.95**	237	37	10577,91	5.93
300*c*−21*s*	12367.6	14229.92	281	49	12594.77	283	46	182.23	12374.49	283	12514.78	4.94	12367.6	283	44	12421.75	47.53	**12235.64**	283	44	12299,83	39.68
350*c*−21*s*	14073.34	16460.3	329	57	14323.02	329	51	232.03	14103.66	329	14271.56	7.11	14073.34	329	50	14226.03	63.01	**13928.79**	329	50	14151,31	24.16
400*c*−21*s*	16660.2	19099.04	378	67	16850.21	378	61	305.12	16697.21	378	16839.23	12.7	16660.2	378	59	17119.89	71.7	**16574.43**	379	58	16655,50	68.92
450*c*−21*s*	18241.48	21854.17	424	75	18521.23	424	68	525.52	18310.6	424	18512.47	13.19	18241.48	424	65	18902.03	80.75	**18183.74**	424	64	18318,29	69.66
500*c*−21*s*	20496.5	24517.08	471	84	21170.9	471	76	356.01	20609.67	471	20874.5	19.51	20496.5	471	73	20997.04	89.95	**20340.27**	471	72	20500,76	65.7
Avg.gap (%)				15.97	1.38				0.17		0.92		0.05			1.02		-0.5			0.19	
Avg.time (%)								159.58				6.2					35.04					24.94

Table 2 provides a comparison between DBCA/MCWS heuristics, VNS/TS, AVNS, MSH and our algorithm. In the case of large instances, our proposed approach perform better results than those provided in the literature for 11 instances among 12 with respect to the solution quality especially our VNS algorithm significantly outperforms DBCA/MCWS algorithms.

5. Conclusion

This work has solved the G-VRP involving the problem of when and where to refuel or recharge the vehicle in order to minimize the cost of the total energy. We have proposed a variable neighborhood search metaheuristic based on a sequential VND algorithm as the local search. The computational results show that our proposed approach provides competitive results with those existing in the literature. Our VNS achieves the best known solutions for all the small and large G-VRP instances and improve results with respect to the solution quality.

References

Bektaş, T., & Laporte, G. (2011). The pollution-routing problem. *Transportation Research Part B: Methodological*, *45*(8), 1232-1250.

Schneider, M., Stenger, A., & Goeke, D. (2014). The electric vehicle-routing problem with time windows and recharging stations. *Transportation Science*, *48*(4), 500-520.

Demir, E., Bektaş, T., & Laporte, G. (2011). A comparative analysis of several vehicle emission models for road freight transportation. *Transportation Research Part D: Transport and Environment*, *16*(5), 347-357.

Demir, E., Bektaş, T., & Laporte, G. (2014). The bi-objective pollution-routing problem. *European Journal of Operational Research*, *232*(3), 464-478.

Erdoğan, S., & Miller-Hooks, E. (2012). A green vehicle routing problem. *Transportation Research Part E: Logistics and Transportation Review*, *48*(1), 100-114.

Fagerholt, K., Laporte, G., & Norstad, I. (2010). Reducing fuel emissions by optimizing speed on shipping routes. *Journal of the Operational Research Society*, *61*(3), 523-529.

Franceschetti, A., Honhon, D., Van Woensel, T., Bektaş, T., & Laporte, G. (2013). The time-dependent pollution-routing problem. *Transportation Research Part B: Methodological*, *56*, 265-293.

Hansen, P., Mladenović, N., & Moreno Pérez, J. A. (2010). Variable neighbourhood search: methods and applications. *Annals of Operations Research*, *175*(1), 367-407.

Hansen, P., Mladenović, N., & Urošević, D. (2006). Variable neighborhood search and local branching. *Computers & Operations Research*, *33*(10), 3034-3045.

Mladenović, N., & Hansen, P. (1997). Variable neighborhood search. *Computers & operations research*, *24*(11), 1097-1100.

Mendoza, J. E., & Villegas, J. G. (2013). A multi-space sampling heuristic for the vehicle routing problem with stochastic demands. *Optimization Letters*, *7*(7), 1-14.

Montoya, A., Guéret, C., Mendoza, J. E., & Villegas, J. G. (2016). A multi-space sampling heuristic for the green vehicle routing problem. *Transportation Research Part C: Emerging Technologies*, *70*, 113-128.

Schneider, M., Stenger, A., & Goeke, D. (2014). The electric vehicle-routing problem with time windows and recharging stations. *Transportation Science*, *48*(4), 500-520.

Schneider, M., Stenger, A., & Hof, J. (2015). An adaptive VNS algorithm for vehicle routing problems with intermediate stops. *OR Spectrum*, *37*(2), 353.