XLVI Symposium on Operational Research
SYM-OP-IS 2019
Kladovo, 15-18 September 2019
UNIVERSITY OF BELGRADE
FACULTY OF ORGANISATIONAL SCIENCES

GENERAL VARIABLE NEIGHBORHOOD SEARCH FOR ASYMMETRIC VEHICLE ROUTING PROBLEM

LUKA MATIJEVIĆ[1], TATJANA DAVIDOVIIĆ[2], VLADIMIR ILIN[2], PANOS M. PARDALOS[3]

[1] Mathematical Institute of the Serbian Academy of Science and Arts, Belgrade, luka@mi.sanu.ac.rs
[2] Mathematical Institute of the Serbian Academy of Science and Arts, Belgrade, tanja@mi.sanu.ac.rs
[3] Faculty of Technical Sciences, Novi Sad, v.ilin@uns.ac.rs
[4] Faculty of Engineering, University of Florida, Gainesville, pardalos@ufl.edu

Abstract: *The minimization of total distance in an asymmetric vehicle routing problem with a hard time window for serving all the customers is considered. This problem is used to model a real-life problem of delivering perishable and non-perishable goods to multiple customers. All the customers should be visited exactly ones with one among a limited number of homogeneous vehicles (with the same capacity and speed). Having in mind that the problem is NP-hard, we developed a General Variable Neighborhood Search (GVNS) approach and tested it on a set of available real-life instances. Our computational results show that the proposed GVNS is able to generate high quality solutions within reasonably short CPU time.*

Keywords: *Combinatorial optimization, Routing of homogeneous vehicles, Single depot, Minimization of total distance, Metaheuristics.*

1. INTRODUCTION

The problem addressed in this paper is how to organize delivery of perishable and nonperishable goods to multiple customers within urban and suburban areas. The problem is widely known in the optimization literature as the Vehicle Routing Problem (VRP) that arises in many everyday situations related to the transportation of goods [5, 16, 17]. In classical VRP, a fleet of v_{max} vehicles need to serve n customers. Each vehicle starts and ends at the depot and each customer must be served exactly once by only one vehicle. There are many variations of VRPs, and for further detailed explanations the readers are referred to [16, 17]. From practical perspective, delivering goods to customers is very complex task which requires holistic and interdisciplinary approach. There are, at least, three key phases that need to be planned and organized in detail. We briefly describe all of them here in order to highlight our specific case.

In the first phase, the information system for an on-line delivery needs to be implemented. Nowadays, electronic business (e-business) has been regularly used for many on-line activities. Electronic commerce (e-commerce) is the most significant portion of e-business due to the possibility to buy or sell goods on-line and make electronic payment as well. However, the final phase of all e-commerce activities is transportation of goods. There are two basic forms of e-commerce: Business-to-Business (B2B) and Business-to-Customer (B2C). The process of transportation planning is different for each of previously mentioned e-commerce forms. In the B2B environment the orders are planned in-advanced, repeated when needed and usually without breakdowns in initial transportation route. In the B2C environment the orders are of small size, instantaneous, and placed by numerous scattered customers [3]. The transportation routes are usually different for each day. In addition, less time is allowed to perform transportation planning in comparison with the B2B environment. In our case customers made orders through a web-site. They need to register, choose the desired goods and depict between two delivery time intervals (between 10AM and 2PM or between 4PM and 8PM) in one of the next four days.

In the next phase, the company needs to organize physical distribution of ordered goods to the customers. The company can own a fleet of vehicles for distribution or it can rent the vehicles for that purpose. Also, the company can own a warehouse for storage of goods or it can rent one. In our case, the company owns a warehouse (distribution center or depot) as well as a fleet of homogenous vehicles with the capacity of 5 tones.

After the on-line orders are made and fleet of vehicles is ready for distribution, the third phase may begin, i.e., transportation planning has to be performed for each day. It is necessary to define a concrete VRP with all required constraints, develop mathematical formulation of the problem, and propose solution methods that will provide optimal or near-optimal solutions. In our case, the classical VRP is extended by three important constraints related to transportation of goods: capacity of vehicles is presumed, one-way or temporarily closed directions within the city are taken into account, and the predetermined time window for delivery is set. The first constraint introduces static and deterministic basic version of the problem in which the demands are known in

advance and cannot be split, the vehicles are identical and are located at the single depot and only the capacity restrictions are imposed (this is marked by a prefix C to VRP). The second constraint introduces asymmetric cost matrix (adds prefix A to CVRP). The third constraint introduces a specific time window, a time period allowed for delivering goods to all the customers (it is denoted by a suffix TW to ACVRP). Therefore, with previously defined constraints we extend the classical VRP problem to Asymmetric Capacitated Vehicle Routing Problem with Time Window (ACVRPTW).

The Mixed-Integer Linear Programming (MILP) formulation for the considered ACVRPTW was proposed in [10] and used within the framework of CPLEX 12.6 commercial solver to solve two real-life test instances. The developed MILP enables to solve to optimality smaller instance within the acceptable running time, while for the larger one the lack of memory problem occurs and a feasible solution with the gap of 18.77% is reported.

Therefore, the next step certainly involves the application of metaheuristics. We decided to use a variant of the well-known Variable Neighborhood Search (VNS) method as, according to the literature, it proved itself very successful for dealing with VRPs [1, 4, 7, 8, 11, 13, 15, 18]. We developed General VNS (GVNS) that explores three neighborhoods, two of them used for shaking and all three in variable neighborhood descent that substitutes the local search phase in GVNS.

The paper is organized as follows. Section 2. contains a brief description of the considered vehicle routing problem (ACVRPTW). Our implementation of GVNS is explained in Section 3., while in Section 4. the experimental evaluation related to the comparison of GVNS and CPLEX commercial solver is presented. Concluding remarks and directions for future work are given in the last section.

2. PROBLEM DESCRIPTION

An asymmetric vehicle routing problem (AVRP) is defined on a complete direct graph $G(N,A)$, where $N = \{0,1,2,\ldots,n,n+1\}$ is the set of vertices (locations, customers) and $A = \{(i,j)|i,j \in N, i \neq j\}$ denotes the set of arcs with weights $d_{i,j}$ representing distances between customers i and j. Vertices 0 and $n+1$ refer to unique location (depot), the origin and destination of each route. This notation is adopted due to the simplicity of formulation.

Each vehicle starts from location 0 (depot), serves a determined subset of customers and then it finishes at location $n+1$ (depot). Each vehicle can perform maximum one tour, while each customer should be served exactly once by only one vehicle. The demands of customers are denoted with c_i, $i = 1,2,\ldots,n$ and they represent the weights of goods to be delivered to each particular customer. The set V of v_{max} homogeneous vehicles (with same capacity Q and same speed s) is provided to serve all the customers' demands. Based on the vehicle speed, the travel time ($t_{i,j}$) between customers i and j can be calculated. The time to serve a customer (to unload goods, deliver them to a customer and collect money) is considered as parameter S_j whose values are given in advance. In order to make the distribution of customers among vehicles more even, the parameter C (representing the maximum number of customers per vehicle) is introduced. Finally, the distribution of goods must be completed during the working hours, and therefore, T denotes the maximum allowed time (time window) for serving customers. The main goal is to minimize the total distance traveled by all vehicles.

3. GVNS IMPLEMENTATION

Variable Neighborhood Search (VNS) is local search based metaheuristic method proposed in [12]. It explores systematic change of neighborhoods, as well as the distances within a single neighborhood, in both a local search phase and for escaping from local optima traps. Basic variant of VNS uses one or more neighborhood structures to explore the search space of a considered optimization problem. It consists of three main steps: Shaking, Local Search, and Move or Not. The role of Shaking step is to prevent an algorithm being trapped in a local optimum. It performs a random perturbation of the current best solution in the given neighborhood. Local Search step has to improve the current solution by examining its neighbors in one or more neighborhoods. After the Local Search phase, the algorithm performs Move or Not step, i.e., it checks the quality of the obtained local optimum and moves there if it is better than the current best one. Otherwise, it just changes the neighborhood for further search. The three main steps are repeated until a pre-specified stopping criterion is satisfied [7, 8].

Among various variants of VNS [6, 9], we implemented General VNS, the variant that explores several neighborhood structures within the Local Search step. Actually, instead of a simple Local Search, a deterministic search through several neighborhoods (referred to as Variable Neighborhood Descent, VND) is used. Our implementation is described in detail in the remainder of this section.

3.1. Solution representation

A solution of our ACVRPTW is represented by v_{max} (where v_{max} is the number of vehicles) doubly linked lists that contain indices of customers assigned to each vehicle in order they are supposed to be visited. We selected this structure because it enables to generate a neighbor of a given solution and to calculate its objective function value in constant number of steps.

3.2. Determination of the initial solution

For generating initial solution we adapted the Nearest Neighbor algorithm, based on [2, 14] in the following way. We start by selecting v_{max} customers closest to the depot and assigning them one to each vehicle. Among the remaining $n - v_{max}$ customers, we select the one that is the closest to the last customer already included in the partial route of one of the vehicles. This customer is appended to the end of the partial route of that vehicle. The process is repeated as long as there are unassigned customers. If the newly generated solution is infeasible, we correct it by some perturbations of customers among vehicles.

3.3. Neighborhood structures

Selecting adequate neighborhood structures is the most important part of the implementation of any VNS metaheuristic. After an extensive testing, we decided to use the following three neighborhood structures [2]:

1. **Insert(k)** ($N_1(k)$) This neighborhood consists of selecting k consecutive customers assigned to one vehicle and moving them to some other position within the route of that same vehicle;
2. **Relocate(k)** ($N_2(k)$) A neighbor in this neighborhood is obtained when k consecutive customers are moved from the currently assigned vehicle to some other vehicle;
3. **Exchange(k)** ($N_3(k)$) The idea here is to select a pair of k consecutive customers from different vehicles and exchange their positions.

Within VND all three neighborhoods are explored, while N_1 and N_2 are used in the shaking phase.

3.4. Shaking

The shaking phase of our GVNS consists of randomly selecting a neighbor at distance k, with respect to either $N_1(\kappa)$ or $N_2(\kappa)$, $\kappa \in \{2,3,4,5\}$ neighborhood, from the current best solution x. Here, k represents the index of the current distance and it changes its values in the range $\{k_{min}, k_{min}+1, \ldots, k_{max}\}$. Neighborhoods are used equally, i.e., each time it is randomly selected which one will be applied. The pseudo-code of the implemented shaking procedure is presented by Algorithm 1.

Algorithm 1 Shaking

```
1: procedure SHAKING
2:     Input: solution x, value of neighborhood index k
3:     i ← 0
4:     x' ← x
5:     while i < k do
6:         i ← i + 1
7:         Randomly select values p ∈ {0,1} and κ ∈ {2,3,4,5}
8:         if p = 0 then
9:             Find random neighbor in N1(κ) of x'
10:        else
11:            Find random neighbor in N2(κ) of x'
       return New solution x'
```

3.5. Variable Neighborhood Descent

Our VND procedure explores all three neighborhood structures of size 1. The order of neighborhoods is determined experimentally: $N_1(1)$, $N_3(1)$, and finally $N_2(1)$. All the neighborhoods are searched according to the First Improvement principle. Neighborhood $N_1(1)$ is explored as long as an improvement could be made. When local optimum according to $N_1(1)$ is reached, the search continues in $N_3(1)$. If an improvement is found

in $N_3(1)$, neighborhood $N_1(1)$ is applied to the newly obtained solution, otherwise, the search continues in $N_2(1)$. An improvement obtained in $N_2(1)$ will direct the search again to $N_1(1)$. On the other hand, if the current solution cannot be improved in $N_2(1)$, VND exits reporting the best obtained local optimum. The pseudo-code of the implemented VND is presented by Algorithm 2.

Algorithm 2 Variable Neighborhood Descent

```
1: procedure VND
2:     Input: solution x
3:     imp ← 1
4:     x' ← x
5:     while imp do
6:         imp ← LS(N1(1), x')
7:         if not imp then
8:             imp ← LS(N3(1), x')
9:         if not imp then
10:            imp ← LS(N2(1), x')
       return New solution x'
```

3.6. The structure of the proposed GVNS

Pseudo-code of the proposed GVNS is presented by Algorithm 3. After all input parameters are read, the initial solution x is determined using the adapted Nearest Neighbor algorithm. It represents also the first estimation of the optimal solution. GVNS then performs search in iterations until the stopping criterion (maximum allowed CPU time) is fulfilled. Within each iteration, k changes between k_{min} and k_{max} and the basic steps (Shaking, Local Search and Move or Not) are performed. Due to a lot of randomness that occurs in the Shaking step, it is hard to control the feasibility of the resulting solution. Therefore, this step is repeated until a feasible solution is obtained. On the other hand, local search examines only the feasible neighbors. At the end, the best obtained solution x is reported. We also measure the CPU time needed to find the best approximation of the optimal solution. It is saved in the variable t_{min} and reported at the end of GVNS execution.

Algorithm 3 GVNS-AVRP

```
1: procedure GVNS
2:     Input: Instance to be solved, k_min, k_min, runtime
3:     Initialization: Determine initial solution x
4:     while Time < runtime do
5:         k ← k_min
6:         while k < k_max do
7:             Shaking:
8:             repeat
9:                 x' ← SHAKING(x, k)
10:            until x' is feasible
11:            Local search: x'' ← VND(x')
12:            Move or Not:
13:            if f(x'') < f(x) then
14:                x ← x''
15:                k ← k_min
16:                t_min ← Time
17:            else
18:                k ← k + 1
       return The best obtained solution x and the CPU time t_min required to find it
```

4. COMPUTATIONAL RESULTS

Our GVNS is coded in C++ programming language and executed on Intel Xeon E5-2620 v3 on 2.40GHz with 31.4 GB RAM and 15 MB cache memory. CPLEX commercial solver is executed on the same platform.

The experimental evaluation is performed on several real-life examples provided by a hypermarket company from Novi Sad, Serbia. The obtained results are summarized in Table 1:. All vehicles have capacity of 5 tones and it is assumed that their average speed is 25 km/h. The stopping criterion for GVNS is set to half an hour, i.e., 1800 s, while CPLEX is allowed to run for 10 hours, i.e., 36000 s.

The structure of Table 1: is as follows. In the first three columns the data describing instances are provided: instance identification, the number of customers, and the number of vehicles. The next three columns are devoted to the results obtained by CPLEX, namely, the objective function value with the gap in the parentheses, the required CPU time, and the time needed to obtain the reported solution. In the case gap equals zero, the optimal solution is obtained. The remaining three columns contain the results provided by GVNS: the best obtained objective function value (with the gap in parentheses), the average objective function value over 10 executions (with the gap in parentheses), and the average required CPU time. As GVNS is a stochastic search method, we performed repetitions with different seed values in order to examine the stability of the proposed method. For every test example, our GVNS was executed 10 times, each time with different seed to initialize random number generator. The gaps are calculated with respect to the CPLEX lower bound. The smaller objective function values and the shorter running times are presented in bold in Table 1:.

Table 1: Comparison of the proposed GVNS and CPLEX

			CPLEX			GVNS		
Example	n	v_{max}	Obj. (% gap)	time [s]	t_{min}	Best obj. (% gap)	Av. obj. (% gap)	Av. time [s]
Ex1	26	2	**50.69(00.00)**	1925.968	300.25	**50.69(00.00)**	**50.69(00.00)**	**10.118**
Ex2	20	3	**20.48(00.00)**	86.121	84.713	**20.48(00.00)**	**20.48(00.00)**	**0.049**
Ex3	25	3	**43.03(00.00)**	5260.865	4312.364	**43.03(00.00)**	**43.03(00.00)**	**0.969**
Ex4	30	3	**46.53(00.00)**	16144.026	598.531	**46.53(00.00)**	**46.53(00.00)**	**23.348**
Ex5	35	3	48.06(3.29)	36000.00	2723.671	**48.04(3.25)**	**48.04(3.25)**	**31.791**
Ex6	45	3	61.18(20.11)	36000.00	35632.153	**59.19(16.20)**	**59.309(16.43)**	**302.063**

As can be seen from Table 1:, GVNS was able to find all optimal solutions provided by CPLEX within several orders of magnitude shorter running times. For the two largest instances, GVNS managed to improve solutions reported by CPLEX. Please note that the presented gaps are related to the CPLEX lower bound, and therefore, cannot guarantee that the solutions obtained by GVNS are not optimal. As can be seen from the times needed by CPLEX to obtain the reported solution (t_{min}), in most of the cases CPLEX was able to find these solutions quite quickly and then it spent a lot of time proving their optimality. Therefore, we can conclude that the proposed GVNS is able to provide high quality solutions within very short execution time which is of vital importance for providing adequate delivery service.

5. CONCLUSION

An asymmetric capacitated vehicle routing problem with time window is considered related to the city delivery service. As our previous results based on the application of CPLEX commercial solver to the developed MILP formulation show, the real-life examples cannot be solved to optimality within given time/memory limits. Therefore, we developed metaheuristic approach based on the variable neighborhood search (VNS) method. The conducted experimental evaluation revealed the superiority of the proposed General VNS (GVNS) over CPLEX with respect to the running time and solution quality for larger test examples. The future research may involve some additional modification of the proposed GVNS, its application to new instances of the considered problem, and the implementation of other metaheuristics.

Acknowledgement

This research was partially supported by Serbian Ministry of Education, Science and Technological Development under the grants nos. OI-174026, OI-174033, and F-159.

REFERENCES

[**1**] de Armas, J. and Melián-Batista, B. Variable neighborhood search for a dynamic rich vehicle routing problem with time windows. *Computers & Industrial Engineering*, 85:120–131, 2015.

[**2**] A. Dhahri, A. Mjirda, K. Zidi, and K. Ghedira. A VNS-based heuristic for solving the vehicle routing problem with time windows and vehicle preventive maintenance constraints. *Procedia Computer Science*, 80:1212–1222, 2016.

[**3**] T. C. Du, E. Y. Li, and D. Chou. Dynamic vehicle routing for online B2C delivery. *Omega*, 33(1):33–45, 2005.

[**4**] H. S. Ferreira, E. T. Bogue, T. F. Noronha, S. Belhaiza, and C. Prins. Variable neighborhood search for vehicle routing problem with multiple time windows. *Electronic Notes in Discrete Mathematics*, 66:207–214, 2018.

[**5**] B. L. Golden, S. Raghavan, and E. A. Wasil, editors. *The vehicle routing problem: latest advances and new challenges*, volume 43. Springer Science & Business Media, 2008.

[**6**] P. Hansen and N. Mladenović. Variable neighborhood search. In *Search methodologies: Introductory Tutorials in Optimization and Decision Support Techniques*, pages 313–337. Springer-Verlag, New York, 2014.

[**7**] P. Hansen, N. Mladenović, J. Brimberg, and J. A. Moreno Pérez. Variable neighborhood search. In *Handbook of metaheuristics*, pages 57–97. Updated edition of a trailblazing volume, Springer, 2019.

[**8**] P. Hansen, N. Mladenović, and J. A. Moreno Pérez. Variable neighbourhood search: methods and applications. *Ann. Oper. Res.*, 175(1):367–407, 2010.

[**9**] P. Hansen, N. Mladenović, R. Todosijević, and S. Hanafi. Variable neighborhood search: basics and variants. *EURO Journal on Computational Optimization*, 5(3):423–454, 2017.

[**10**] V. Ilin, L. Matijević, T. Davidović, and P. M. Pardalos. Asymmetric capacitated vehicle routing problem with time window. In *Proc. XLV Symposium on Operations Research, SYM-OP-IS 2018*, pages 174–179, Zlatibor, Serbia, 2018.

[**11**] S.-C. Liu and A.-Z. Chen. Variable neighborhood search for the inventory routing and scheduling problem in a supply chain. *Expert Systems with Applications*, 39(4):4149–4159, 2012.

[**12**] N. Mladenović and P. Hansen. Variable neighborhood search. *Comput. & OR*, 24(11):1097–1100, 1997.

[**13**] M. Schneider, A. Stenger, and J. Hof. An adaptive vns algorithm for vehicle routing problems with intermediate stops. *OR Spectrum*, 37(2):353–387, 2015.

[**14**] M. M. Solomon. Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations research*, 35(2):254–265, 1987.

[**15**] K. Sörensen, M. Sevaux, and P. Schittekat. "Multiple Neighbourhood" Search in Commercial VRP Packages: Evolving Towards Self-Adaptive Methods. In *Adaptive and multilevel metaheuristics*, pages 239–253. Springer, 2008.

[**16**] P. Toth and D. Vigo, editors. *The vehicle routing problem*. SIAM, 2002.

[**17**] P. Toth and D. Vigo, editors. *Vehicle Routing: Problems, Methods, and Applications*. MOS-SIAM Series on Optimization. SIAM, second edition edition, 2014.

[**18**] N. Wassan, N. Wassan, G. Nagy, and S. Salhi. The multiple trip vehicle routing problem with backhauls: Formulation and a two-level variable neighbourhood search. *Computers & Operations Research*, 78:454–467, 2017.