An Augmented Variable Neighborhood Search to Solve the Capacitated Location-Routing Problem

Alireza Mohamadi-Shad, Hamed Niakan, and Hasan Manzour

Abstract — In this paper we proposed a new variable neighborhood search (VNS) for solving the location-routing problem with considering capacitated depots and vehicles. A set of capacitated vehicles, a set of depots with restricted capacities, and associated opening costs, and a set of customers with deterministic demands are given. The problem aims to determine the depots to be opened, fleet assignment to each depot, and the routes to be performed to satisfy the demand of the customers. The objective is to minimize the total costs of the open depots, the setup cost associated with the used vehicles, and transportation cost. We proposed a new VNS which is augmented with a probabilistic acceptance criterion as well as a set of efficient local searches. The computational results implemented on four well-known data sets demonstrate that the proposed algorithm is competitive with other well-known algorithms while reaching many best-known solutions and updating six best new results with reasonable computational time. Conclusions and future research avenues close the paper.

Index Terms — Capacitated location routing, variable neighborhood search, heuristic, local search, simulated annealing, dynamic penalty function.

I. INTRODUCTION

In this paper, we focus on Capacitated Location-Routing Problem (CLRP) where restriction on both depots and vehicles in terms of capacity is taken into account. The CLRP stated in this paper can be represented on a complete, weighted, and undirected network $G=(V, E)$. $V=I \cup J$ includes a set of m potential nodes for depots indexed by $I=\{1, 2, \dots, m\}$ and a set n customers indexed by $J=\{1, 2, \dots, n\}$. Each depot, $i \in I$, is characterized by a limited capacity and associated fixed cost. Each customer, $j \in J$, has a non-negative predefined demand which should be satisfied. We assume that using a vehicle (i.e., starting a route) incurs a fixed cost, and there is a travel cost to move from node i to node j . The goal is to determine which depots need to be opened, how to allocate vehicles to the depots, and how to dispatch vehicles to fulfill customers so as the total system cost is minimized without violating the constraints.

The total system cost encompasses three components including the opening cost of established depots, fixed cost of incurring vehicles (cost of starting a tour), and the total

transportation cost. The following constraint must be met (i) each customer must be assigned to one and only one vehicle (ii) each vehicle must begin and end at the same depot and its total load must not exceed vehicle capacity (iii) the total demand of customers allocated to a depot must not exceed capacity of the depot. Note that CLRP is a NP-hard problem, as it encompasses two NP-hard problems (facility location and vehicle routing).

LRP has received a great deal of attention in literature. Interested reader can refer to Prodhon and Prins [20], Lopes et al. [10], and Nagy and Salhi [14] as valuable surveys for LRP. We briefly mention those studies on LRPs which are closely related to ours i.e., CLRP. Perl and Daskin [15], [16] develop a sequential algorithm to solve CLRP in which the problem is decomposed into three sub-problems. Later, the proposed method is modified by Hansen et al. [8], and Wu et al. [23]. Tuzun and Burke [22] develop a two-phase tabu search heuristic to cope with this problem. Prins et al. [18] propose a metaheuristic wherein Greedy Randomized Adaptive Search Procedure (GRASP) is executed in the first phase, based on an extended and randomized version of Clarke and Wright's algorithm. In the second phase, path relinking as a post optimization is used to enhance the results even further. Prins et al. [17] hybrid a memetic algorithm with population management (MA|PM) to tackle the problem. Prins et al. [19] combine the Lagrangian relaxation technique with Granular Tabu Search (GTS) in an iterative manner to solve the CLRP. The algorithm alternated between a depot location phase and a routing phase, exchanging information on the most promising edges. Barreto et al. [3] develop a sequential method to solve the problem. They firstly clustered customers according to the capacity of vehicles, and then solve a TSP for each cluster. Then, they assign tours to depots. Marinakis and Marinaki [11] combine a particle swarm optimization (PSO) algorithm, greedy randomized adaptive search procedure (MPNS-GRASP) algorithm, the expanding neighborhood search (ENS) strategy and a path relinking (PR). Marinakis et al. [12] modified their approach by using honeybees mating optimization (HBMO) algorithm instead of PSO. Yu et al. [24] introduce a hierarchical heuristic algorithm based on Simulated Annealing (SA) algorithm. In order to better explore the solution space, they use a special encoding scheme and three neighborhood structures. Duhamel et al. [4] integrate a GRASP and Evolutionary Local Search (ELS), called GRASP×ELS, for solving CLRP. They tested their proposed algorithms on three different well-known benchmark instances with or without limited capacities on depot. Escobar et al. [5] develop a two-phase hybrid heuristic algorithm (2-Phase HGTS) to tackle the CLRP. In the first phase, known as Construction phase, a hybrid

Submitted on February 23, 2021.

Published on June 10, 2021.

Alireza Mohamadi-Shad, Department of Industrial Engineering, Bu-Ali Sina University, Iran.

Hamed Niakan, Department of Industrial & Systems Engineering, Wayne State University, USA.

(e-mail: hniakan@wayne.edu)

Hasan Manzour, Department of Industrial and Systems Engineering, University of Washington, USA.

method capable of generating good initial solution is developed. Then, during the second phase, i.e., intensification phase, a modified granular tabu search along with a random perturbation procedure is applied. Ting and Chen [21] apply a hierarchical Multiple Ant Colony Optimization algorithm (MACO) to solve CLRP. Fazeli et al. [7] introduced a min-max autonomous vehicle routing problem which seems promising for solving such transportation network problems and determining the most efficient routing strategies. In his other paper, Fazeli et al. [6] proposed a choice modeling approach embedded in a two-stage stochastic programming mode which is very practical to efficiently solve big network/optimization problem. Although VNS has been claimed to be a really effective metaheuristic for combinatorial optimizations Mladenovic' and Hansen [13], it has not been applied for CLRP yet. However, Jarboui et al. [9] embed a variable neighborhood descent as the local search into the general variable neighborhood heuristic framework to solve the LRP with uncapacitated vehicles and obtained promising results. Hence, the main aim of the paper is to develop an augmented variant of VNS suitable for CLRP.

The rest of this paper is outlined as follows. The proposed solution approach is presented in Section 2. Computational results are presented and analyzed in Section 3, and finally we close the paper by providing some future research avenues.

II. SOLUTION FRAMEWORK

The proposed algorithm to solve the CLRP is built based upon Variable Neighborhood Search (VNS) introduced by Mladenovic' and Hansen [13]. In order to diversify the generated solutions, a simulated annealing acceptance criterion is embedded into the framework. Moreover, a dynamic penalizing strategy allowing the search to be oscillated between feasible and infeasible part of solution space is introduced to avoid trapping into the local optima. A new local search in addition to some other local searches are also embedded into the proposed VNS.

The VNS is a simple but effective approach. The main idea of VNS comes from this fact that a global optimum is actually the best local optima. Thus, VNS aims to alter the neighborhood whenever a local optimum is obtained. Hence, VNS is principally a local search tool that has the capability of escaping from trapping into local optima by searching a set of different neighborhoods to find a better local optima. At the next subsection, we briefly outline the VNS structure. Another innovative but very practical approach for solving capacitated location-routing problem is presented by Ansari-pour and Trafalis [1]. They used a robust optimization model with box constraints for city logistic terminal locations. This approach can be useful to manage the uncertain demand in the market!

A. The Basic of VNS

The VNS composes three basic steps: i) shaking step, that is a stochastic phase which finds a random neighbor of the incumbent solution, ii) local search phase, which aims to find a local optimum usually in a deterministic fashion phase, and iii) the acceptance step, which admits a solution

that improves the objective. Practically, implementing the VNS algorithm requires to (1) find an initial solution, (2) define a set of neighborhoods (shaking phase) (3) define a criterion to determine when to perform change of neighborhood and (4) define the method to explore each of them (local search phase).

Suppose that there are k_{max} neighborhood structures $k = \{1, 2, \dots, k_{max}\}$ such that N_k is the k th neighborhood type. VNS begins with an initial solution, S (current local optima). VNS encompasses two main loops. The former is the shaking loop with which solution S moves from one point to a neighborhood point, say S' . In the latter one, the local search phase (internal loop), the solution S' is improved based on some local searches or a defined mechanism to reach to a local optima, say S^* . After terminating the local search phase, the obtained local optima are compared to the S i.e., the best local optima so far. If S^* is better than S , S^* is substituted with S (Move-or-Not phase) and the search reverts back to shaking phase, and the first neighborhood search applies. Otherwise, the next neighborhood search, i.e., N_{k+1} , in the shaking phase is applied on S . The algorithm is terminated when k exceeds k_{max} .

```

Input: The set of neighborhood structures  $N_k$  for  $k = 1, 2, \dots, k_{max}$ 
Step (0) Initialization: Find an initial solution  $S$ 
while  $k \leq k_{max}$  do
Step (1)  $S' \leftarrow$  a random neighborhood in  $N_k(S)$  **Shaking Phase**
Step (2)  $S^* \leftarrow$  Local search ( $S'$ ) **Local search phase**
Step (3)
    if  $f(S^*) < f(S)$  then  $S \leftarrow S^*$  ** Move-or-not phase**
    else
         $k \leftarrow k + 1$  ** Neighborhood change**
Return  $S$ 

```

Fig. 1. Standard of VNS.

B. An Overview of the Proposed Algorithm

The proposed algorithm begins with an initial solution, say $S = S_0$. A random solution is chosen initially. Then, the main loop of the algorithm (i.e., shaking loop) starts with the first neighborhood structure i.e., $k = 1$, and the initial solution is converted to S' . The set of neighborhood searches is defined later. Next, the internal loop (i.e., local search loop) is applied to find a local optimum, say S^* . In contrast to the classical VNS where S is substituted with S^* ($S = S^*$) only when S^* produces better result, in this paper, we exploit a simulated annealing criterion to determine when to accept a solution. In fact, an inferior solution in move-or-not phase can be chosen according to the new defined acceptance criteria. This enables the algorithms to explore the solution space more. The main steps of our algorithm are given in Fig 2, but the explanations of these steps are elaborated in the following subsections.

C. Explanation of the Main Steps

1. Solution representation and cost function

In this paper, we used a simple solution representation. An array with the length of L is used to encode a solution. The set of indices $\{1, 2, \dots, n\}$ denotes the customers, the set of indices $\{n + 1, n + 2, \dots, n + m\}$ denotes the depots and a set of dummy zeros is used to separate routes belonging to the same depots. As illustrated in Fig. 3, we can assume an instance with 7 customers and 3 candidate depots. In this example, the first route starts from depot 11 and serves three customers (i.e., 3, 5, 7) and goes back to its depot. The

second tour associated with depot 11 fulfills the second and first customers, respectively. The depot 12 is closed as there is no customer between depot 12 and 13. Similarly, customers 8,6 and 10,9 are fulfilled by the depot 13 using two separate routes. An illustration is given in Fig. 4. Similar encoding scheme is used. The main benefit of this encoding is to be capable of producing very diverse solutions which helps to explore solution space whereas it may increase the computational time.

Step (0) Initialization. Define a set of neighborhood structures $N_k, k = \{1, 2, \dots, k_{max}\}$, and set the value for T_0, T_f, K_{max} .

Step (1) Generate an initial solution S randomly and set $S^* = S$, and $T = T_0$.

Step (2) repeat the following steps till $T \leq T_f$

Step (3) repeat the following steps till $k \leq k_{max}$

3-1) **Shaking phase:** $S' \leftarrow$ a random neighborhood in $N_k(S)$

3-2) **Local search phase:** $S^* = local_search(S')$

3-3) **Move or not phase:** $\Delta = f(S^*) - f(S)$
If $\Delta \leq 0$ $S \leftarrow S^*$; and $k \leftarrow k + 1$
Otherwise, $k \leftarrow k + 1, r = uniform(0, 1)$. If $(r < exp(-\Delta/T))$ $S \leftarrow S^*$

3-4) Return to Step (3)

Step (5) Updating Temperature $T = T \times \alpha$.

Step (6) $\beta_1 \leftarrow \beta_1 \cdot (1 + \delta_1)$, $\beta_2 \leftarrow \beta_2 \cdot (1 + \delta_2)$ and revert back to Step (2)

Fig. 2. The overall framework of the proposed VNS.

Fig. 4. Illustration of the proposed VNS.

The proposed solution representation can correspond to a solution that violates the problems constraints i.e., vehicle and depot capacity. Although a solution should not remain in a space which violates the problem constraints, it can explore the infeasible region with the aim of finding promising local optima located near to the infeasible region. A dynamic penalty function which smoothly increases penalty coefficient associated with depot and vehicle capacity violations to shift algorithm from diversification (i.e., global search) of whole space to intensification (i.e., local search) of feasible space is also embedded.

Mathematically speaking, the cost function associated with a solution, say S can be obtained by the following formula:

$$\eta(S) = C(S) + \beta_1 \cdot \xi_1(S) + \beta_2 \cdot \xi_2(S)$$

where $C(S)$ is the value of the objective function. ξ_1, ξ_2 are the value of the violations associated with the vehicle and depot capacity, respectively. β_1, β_2 are the penalty coefficients corresponding to one unit violation of vehicle and depot capacity, respectively. At step (6), these coefficients are updated after changing the temperature with the factor of $(1 + \delta_1)$ and $(1 + \delta_2)$, respectively. In addition, one can use the new techniques presented by Ansaripour et al. [2], to deal with the stochastic aspects of the problem when it arises. This approach would be very helpful to find a close form expression for the cumulative distribution function of the maximum value of the objective function which is not very easy to solve by other existing solutions in the literature!

2. Shaking phase (Step 3-1 of Fig. 3)

The proposed algorithm utilizes four neighborhood structures (i.e., $k_{max} = 4$) for both shaking and local search phases. The first three local searches are used within the domain of VRP while the last one is introduced in this paper. The order of the neighborhoods which is chosen after some experimentation is as follows: the relocation is used as N_1 , the SWAP is used as N_2 , the inversion is chosen as N_3 , and Combined Relocation and SWAP move (CRS) is the last neighborhood search.

In the first neighborhood search, two random numbers are generated e.g., i, j and the i th element is removed from its position and inserted immediately after the j th element of the solution representation. As illustrated in Fig. 5a, two elements 10 and 12 (i.e., 4, 6 customers) are chosen, and the fourth customer is removed and inserted after the sixth customer. In the next neighborhood search, known as SWAP, two points are randomly chosen, and the position of those are inter-changed. As illustrated in Fig. 5b, the positions of fifth and seventh elements are changed. The inversion neighborhood search selects two elements randomly and then change the positions of those, and also inverse the sequence of all elements among those. As illustrated in Fig. 5c the fourth and ninth elements are changed and the order of elements among those are reversed. In the last neighborhood search, CRS, at first two elements, say i, j , are randomly chosen. In addition, two random numbers μ_1, μ_2 in the interval $[0, k]$ are randomly selected. The selected random number must satisfy these constraints $i + \mu_1 \leq j, j + \mu_2 \leq L$. Then, the position of i th element and next μ_1 elements after i are altered with the position of j th element as well as μ_2 elements after j . For instance, in Fig. 5d-1, $i = 3, j = 9, \mu_1 = 0, \mu_2 = 3$. The Fig. 5d-2 shows another example where $i = 3, j = 9, \mu_1 = 2, \mu_2 = 2$. In this local search, if $\mu_1 = 0, \mu_2 = 0$, the local search reduces to SWAP move.

11	3	5	7	0	2	1	12	13	4	8	6	0	10	9	0	0
----	---	---	---	---	---	---	----	----	---	---	---	---	----	---	---	---

Fig. 3. Solution encoding.

a															
11	3	5	7	0	2	1	12	13	<u>4</u>	8	<u>6</u>	0	10	9	0
11	3	7	0	2	1	5	12	13	8	<u>6</u>	<u>4</u>	0	10	9	0
b															
11	3	5	7	<u>0</u>	2	<u>1</u>	12	13	4	8	6	0	10	9	0
11	3	5	7	<u>1</u>	2	<u>0</u>	12	13	4	8	6	0	10	9	0
c															
11	3	5	<u>7</u>	0	2	1	12	<u>13</u>	4	8	6	0	10	9	0
11	3	5	<u>13</u>	<u>12</u>	<u>1</u>	<u>2</u>	<u>0</u>	<u>7</u>	4	8	6	0	10	9	0
d-1															
11	3	<u>5</u>	7	0	2	1	12	<u>13</u>	<u>4</u>	<u>8</u>	6	0	10	9	0
11	3	<u>13</u>	<u>4</u>	<u>8</u>	6	7	0	2	1	12	<u>5</u>	0	10	9	0
d-2															
11	3	<u>5</u>	7	0	2	1	12	<u>13</u>	4	8	6	0	10	9	0
11	3	<u>13</u>	4	8	2	1	12	<u>5</u>	7	0	6	0	10	9	0

Fig. 5. (a) relocation move. (b) SWAP move. (c) Inversion move. (d-1,2) CRS move.

3. Local search phase (Step 3-2 of Fig. 3)

After the shaking phase, we need to take into account the local search phase. In the local search phase, the proposed algorithm uses a simple and straightforward procedure to avoid extensive computational time. That is, the algorithm is devised such that the local search phase takes advantage of the same neighborhood structures used in the shaking phase. The local search phase is repetitively performed until there is no improvement in I consecutive iterations. The procedure is presented in Fig. 6.

Step 1) Define the set of neighborhood structures, and the maximum number of successive iterations without improvement, say I
Step 2) S is the neighborhood solution of an iteration
Step 3) repeat the following steps while $Iter \leq I$
 $S'_k = N_k(S_k)$
 If $f(S'_k) < f(S_k)$
 $S_k = S'_k, Iter = 0$
 else $iter \leftarrow iter + 1$
Step (4) return S_k as a local minimum.

Fig. 6. Local search phase of the proposed algorithm.

4. Simulation Annealing Criterion (Step 3-3 of Fig. 3)

The cost function of the obtained local optima in classical VNS is non-increasing. In this paper, however, we embed a probabilistic criterion enabling the algorithm to accept a local minimum even though it is inferior compared to the best obtained local minimum. In fact, if obtained local minimum is better than the best local minimum, it is substituted. Then, the algorithm reverts back to the main loop and resumes from the first neighborhood structure. Otherwise, making decision about substitution is according to a probability function. Thus, $S = S^*$ if $r < \exp(-\Delta/T)$ where r is a random number between (0,1), and Δ is absolute cost between S and S^* , T is the current system temperature, and the search starts from the first neighborhood. Otherwise, the next neighborhood structure is applied on S and the local search phase is performed again. This process is repetitively done till $k \leq k_{max}$. The VNS algorithm is done for a specific temperature, then the temperature is updated and the whole VNS algorithm restarts.

The main philosophy of this approach is that decreasing the objective function monotonically increases the chance of trapping into a local optimum. Thus, by relaxing this constraint and providing this flexibility for the algorithm to accept a worst solution according to a probability distribution (i.e., simulated annealing acceptance criterion), we can increase the chance of enhancing the results even further.

In addition, a monotonically increasing coefficient of penalty function enables algorithms to explore the solution space wisely without extensive computational time.

III. COMPUTATIONAL RESULTS

In this section, the computational results of the proposed VNS algorithm are presented. The proposed VNS algorithm is coded in Visual C++ and run on a Core 2 duo processor at 2.00 GHz with 1 GB of RAM under Windows 7 operating system. A set of preliminary experiments are performed to determine a proper set of parameters producing satisfactory results across most instances even though they are not optimal for all instances. According to these experiments, the parameters are set as follows: $T_0 = 50$, $T_f = 0.001$, $\alpha = 0.92$, $I = L$, $(L - 1)$, $\beta_1 = 5$, $\beta_2 = 10$, $\delta_1 = \delta_2 = 0.02$. Please note that L refers to the length of array associated with the initial solution.

We implemented the proposed Hybrid VNS (HVNS) on three different set of benchmark instances from the literature, (i) Prodon's instances, (ii) Barreto's instances, (iii) Tuzan and Burke's data sets, containing a total of 79 instances, with the same parameter settings.

In order to assess the effectiveness of the proposed algorithm, it is compared with seven of the most recently published algorithms: The greedy adaptive randomized search procedure (GRASP) developed by Prins et al. [18], the memetic algorithm with population management (MA|PM) by Prins et al. [17], the lagrangean relaxation granular tabu search (LRGTS) proposed by Prins et al. [19], the heuristic simulated annealing algorithm (SALRP) developed by [24], the GRASP hybridized with an

evolutionary local search algorithm (GRASP×ELS) introduced by Duhamel et al. [4], two-phase hybrid heuristic algorithm (2-Phase HGTS) developed by Escobar et al. [5], and the multiple ant colony optimization algorithm (MACO) introduced by Ting and Chen [21].

The first five proposed algorithms are robust, and the authors just ran one run. The robustness means that the solution obtained by these algorithms has relatively similar performance over multiple runs. Nevertheless, Duhamel et al. [4] performed 5 runs for GRASP×ELS and Ting and Chen [21], performed 10 runs for MACO. In fact, the multiple runs could improve the quality of the obtained results. In this paper, we provided both single run and multiple run (10 runs) to provide comprehensive and fair results in comparison to all other proposed algorithms.

The results are elaborated in Tables I-VII. Two tables are given for each data set, and for the Prodhon's instances, we added another table to show the contribution of each component of the proposed solution. The first table of each data set is associated with the result of the proposed method and the best given solutions in the literature, and the second reports comparison results. In the first table of each data set, the columns show the name of instances (Problem ID), number of customers (n), number of depots (m), capacity of vehicles (Q), the best given solution in the literature (BKS), objective function for 1 and 10 runs (cost), corresponding computational time (CPU) and gap (GAP), respectively. The GAP is calculated by:

$$GAP = \frac{C - BKS}{BKS} \times 100$$

where C is the associated cost value. The last column reports the Coefficient of Variation (CV) that is a good indicator of robustness of the proposed algorithm. The second table of each data set composes the problem ID and GAP associated with the seven aforementioned proposed algorithms in the literature in addition to the proposed algorithm in this paper. Moreover, the last two rows denote the average gap and the number of best solutions obtained by each method for each method. If the obtained solution is the same as BKS or better than the best-known solution, we bold it in the Tables I, III, and V. In addition, the best obtained solutions among the proposed algorithm and the considered seven other algorithms are bold in Tables II, IV, and VI. Note that the BKS does not necessarily come from these seven last published papers or the proposed algorithm in this paper. The detail information about each data set is elaborated as follows.

A. Prodhon's Instances

The Prodhon's set contains 30 LRP instances with capacitated routes and depots. The complete data sets are available at <http://prodhonc.free.fr/>. One attribute of this data set is that it contains the largest instances with capacitated depots (10 depots and 200 customers). The number of depots is either 5 or 10, the number of customers is chosen from the set {20, 50, 100, 200} and vehicle capacities are either 70 or 150. The vehicle fixed cost is considered in this group of instances. Moreover, another attribute of this data set is that it contains both uniform and

clustered customers distribution (either two clusters or three). The demand of customers is generated in a uniform distribution in interval [10], [20].

Table I summarizes the computational results of the proposed algorithm associated with Prodhon's instances. Table II is dedicated to the comparison results. Table III reports the contribution of each component of the proposed VNS in refining the objective function. According to Table I, the average gaps associated with single and multiple runs are 0.54 and 0.04. The single and multiple runs HVNS reaches 11 and 24 over 30 instances of best-known solutions in the literature respectively. In addition, the HVNS with single run updated the solution of one instance (i.e., 200-10-1b) and multiple runs HVNS updates the BKS for 5 instances. The obtained results clearly show the high quality performance of the proposed method in terms of generating promising results.

According to Table II, the proposed multiple runs HVNS reaches 27 over 30 best solutions among these 7 tested algorithms. Quite evidently, the proposed algorithm dominates all other previous algorithms in terms of both average solution quality and the number of best solutions. The proposed algorithm did not find the best results only in a few cases with negligible gap. In respect to single run, the proposed algorithm is competitive with those papers with multiple runs (i.e., 2-Phase HGTS and MACO) while dominates all other single run algorithms in terms of average gap.

As can be seen in Table III, the value of adding different components of the proposed algorithm is reported. The GAPs between the HVNS and the HVNS without SA, new local search, and penalizing strategy are reported in Table III. It indicates that all components have had significant impact on improving the quality of the solution. The SA seems to be most important component while the local search and penalizing strategy are the next important components, respectively.

Since the computational time depends on many factors (e.g., machines, compiler, programming language, etc.) it is difficult to compare the run time directly. However, we would like to emphasize that the machine we used is at least 3 times slower than the one used by Ting and Chen [21], Escobar et al. [5]. Considering this fact, the proposed algorithm is competitive (e.g., Ting and Chen [21], Escobar et al. [5]) in terms of computational time.

<p>SALRP: PC with an Intel Core 2 Quad CPU (2.6 GHz) and 2 GB memory</p> <p>2-phase HGTS: Intel Core Duo CPU (2.00 GHz) under Linux Ubuntu 11.04 with 2 GB of memory</p> <p>MACO: runs on a PC with an Athlon XP 2500+ (1.83 GHz) processor and 512 MB RAM, under the Windows XP operating system</p> <p>GRASP, GRASP×ELS, LRGS, MAPM: on a quad core 2.83 GHz computer under Windows XP with 8 GB of memory.</p> <p>HVNS: core2 duo 2.00 GHz and 1 GB of RAM.</p>
--

(Dr. Dullaert, this is for your information)

TABLE I: PRODHON'S DATA SET, COMPARISON WITH BKS

problem ID	n	m	Q	BKS	Cost		CPU		GAP		C. V
					1 run	10 run	1 run	10 run	1 run	10 run	
20-5-1a	20	5	70	54,793	54,793	54,793	19	25	0.00	0.00	0.000
20-5-1b	20	5	150	39,104	39,104	39,104	27	22	0.00	0.00	0.000
20-5-2a	20	5	70	48,908	48,908	48,908	30	33	0.00	0.00	0.000
20-5-2b	20	5	150	37,542	37,542	37,542	32	38	0.00	0.00	0.000
50-5-1a	50	5	70	90,111	90,111	90,111	43	39	0.00	0.00	0.000
50-5-1b	50	5	150	63,242	63,242	63,242	28	40	0.00	0.00	0.000
50-5-2a	50	5	70	88,298	88,643	88,298	51	46	0.39	0.00	0.008
50-5-2b	50	5	150	67,308	67,308	67,308	38	37	0.00	0.00	0.000
50-5-2bis	50	5	70	84,055	84,055	84,055	43	33	0.00	0.00	0.000
50-5-2Bbis	50	5	150	51,822	52,059	51,822	41	48	0.46	0.00	0.007
50-5-3a	50	5	70	86,203	86,203	86,203	65	54	0.00	0.00	0.000
50-5-3b	50	5	150	61,830	62,061	61,830	19	31	0.37	0.00	0.006
100-5-1a	100	5	70	274,814	277,513	275,066	174	245	0.98	0.09	0.009
100-5-1b	100	5	150	213,615	215,068	213,618	198	164	0.68	0.00	0.005
100-5-2a	100	5	70	193,671	194,123	193,671	143	176	0.23	0.00	0.002
100-5-2b	100	5	150	157,095	158,002	157,129	102	114	0.58	0.02	0.002
100-5-3a	100	5	70	200,079	200,146	200,079	254	182	0.03	0.00	0.003
100-5-3b	100	5	150	152,441	153,127	152,441	241	282	0.45	0.00	0.008
100-10-1a	100	10	70	287,983	290,726	288,843	204	166	0.95	0.30	0.004
100-10-1b	100	10	150	231,763	231,888	231,028	351	243	0.05	-0.32	0.004
100-10-2a	100	10	70	243,590	243,590	243,590	141	167	0.00	0.00	0.003
100-10-2b	100	10	150	203,988	205,627	203,988	119	183	0.80	0.00	0.006
100-10-3a	100	10	70	250,882	251,725	250,882	307	223	0.34	0.00	0.007
100-10-3b	100	10	150	204,317	207,871	204,188	146	191	1.74	-0.06	0.007
200-10-1a	200	10	70	476,778	495,979	476,516	1094	828	4.03	-0.05	0.019
200-10-1b	200	10	150	378,289	377,809	377,809	888	974	-0.13	-0.13	0.021
200-10-2a	200	10	70	449,571	451,623	449,470	1260	1120	0.46	-0.02	0.028
200-10-2b	200	10	150	374,330	377,625	375,159	994	1092	0.88	0.22	0.016
200-10-3a	200	10	70	469,433	476,743	472,962	954	891	1.56	0.75	0.003
200-10-3b	200	10	150	362,817	367,615	364,737	1361	1276	1.32	0.53	0.004
Ave.							312	299	0.54	0.04	0.006

TABLE II: PRODHON'S INSTANCES, COMPARISON WITH OTHER ALGORITHMS

Problem ID	GRASP		MA M		LRGTS		SALRP		GRASP*ELS		2-Phase HGTS		MACO		HV S	
	GAP	CPU	GAP	CPU	GAP	CPU	GAP	CPU	GAP	CPU	GAP	CPU	GAP	CPU	GAP	CPU
20-5-1a	0.42	0	0.00	0	0.62	0	0.00	20	0.00	0	0.00	3	0.00	4	0.00	25
20-5-1b	0.00	0	0.00	0	0.00	0	0.00	15	0.00	0	0.00	4	0.00	5	0.00	22
20-5-2a	0.00	0	0.00	0	0.00	1	0.00	19	0.00	0	0.08	3	0.00	4	0.00	33
20-5-2b	0.00	0	0.00	0	0.00	0	0.00	15	0.00	0	0.00	4	0.00	5	0.00	38
Ave.	0.10	0	0.00	0	0.15	0	0.00	17	0.00	0	0.02	4	0.00	5	0.00	30
50-5-1a	0.58	2	0.05	3	0.05	0	0.00	75	0.00	3	0.32	27	0.00	24	0.00	39
50-5-1b	2.37	2	0.00	3	0.02	1	0.00	58	0.00	0	1.31	27	0.00	21	0.00	40
50-5-2a	0.55	2	0.00	3	0.47	2	0.00	95	0.39	11	1.18	23	0.00	24	0.00	46
50-5-2b	1.09	3	0.87	3	0.58	2	0.05	59	0.00	16	1.74	21	0.00	20	0.00	37
50-5-2bis	0.00	2	0.00	3	0.15	2	0.00	75	0.00	0	0.00	23	0.00	25	0.00	33
50-5-2Bbis	0.46	3	0.00	4	0.33	1	0.00	66	0.00	11	0.51	29	0.00	17	0.00	48
50-5-3a	1.37	2	0.00	3	0.00	0	0.29	74	0.00	0	0.00	66	0.00	33	0.00	54
50-5-3b	0.10	2	0.00	5	0.00	1	1.41	58	0.00	0	0.00	38	0.00	26	0.00	31
Ave.	0.81	2	0.12	3	0.20	1	0.22	70	0.05	5	0.63	32	0.00	24	0.00	41
100-5-1a	1.68	28	2.59	26	1.14	9	0.81	349	0.78	148	0.50	157	0.51	117	0.09	245
100-5-1b	1.19	23	1.42	35	0.59	8	1.12	269	1.05	68	0.60	136	0.33	135	0.00	164
100-5-2a	3.02	17	0.98	36	1.49	2	0.23	349	0.31	212	0.49	145	0.40	237	0.00	176
100-5-2b	1.56	22	0.15	36	0.44	3	0.04	212	0.18	125	0.14	193	0.08	144	0.02	114
100-5-3a	1.96	22	0.83	29	0.94	2	0.08	250	0.13	141	0.50	163	0.48	179	0.00	182
100-5-3b	1.41	20	0.58	33	1.49	3	0.02	197	0.06	221	0.80	168	0.18	152	0.00	282
Ave.	1.80	22	1.09	33	1.01	5	0.38	271	0.42	153	0.51	160	0.33	161	0.02	194
100-10-1a	12.22	37	9.93	25	1.36	14	1.06	270	4.67	48	0.62	277	1.09	105	0.30	166
100-10-1b	17.14	30	16.61	36	1.63	14	1.06	203	16.32	186	2.69	152	1.55	82	-0.32	243
100-10-2a	4.31	39	0.63	25	1.28	14	0.91	261	0.08	260	0.89	92	0.69	122	0.00	167
100-10-2b	1.26	30	0.52	32	0.22	10	0.65	199	0.00	139	0.13	99	0.75	85	0.00	183
100-10-3a	7.95	35	1.11	29	3.10	13	0.00	338	1.05	164	1.53	125	1.36	112	0.00	223
100-10-3b	5.80	40	0.24	37	0.77	11	0.34	240	0.38	203	0.74	144	0.23	80	-0.06	191
Ave.	8.11	35	4.84	30	1.39	13	0.67	252	3.75	167	1.10	148	0.95	98	-0.01	196
200-10-1a	2.95	518	1.41	345	1.03	62	0.89	1428	2.03	1521	0.00	671	0.43	942	-0.05	828
200-10-1b	10.17	379	0.46	463	0.61	60	1.40	1336	1.07	359	0.00	476	0.15	562	-0.13	974
200-10-2a	14.04	554	0.50	281	0.84	60	0.28	1796	0.60	112	0.08	483	0.42	704	-0.02	1120
200-10-2b	1.51	367	0.18	321	0.81	77	0.63	1245	0.45	1610	0.17	530	0.17	404	0.22	1092
200-10-3a	5.81	425	1.85	213	1.54	77	0.95	1776	1.91	1596	0.62	624	1.22	879	0.75	891
200-10-3b	7.22	290	0.56	272	0.67	73	0.24	1326	0.65	591	0.12	389	0.71	491	0.53	1276
Ave.	6.95	422	0.83	316	0.92	68	0.73	1485	1.12	965	0.16	529	0.52	663	0.22	1030
Global Ave.	3.60	96	1.38	77	0.74	17	0.41	422	1.07	258	0.53	176	0.36	191	0.04	299
Num. of Best	4		10		5		10		12		9		13		27	

TABLE III: CONTRIBUTION OF EACH COMPONENT OF VNS

problem ID	HVNS		HVNS without SA			HVNS without new local search			HVNS without dynamic penalizing		
	Cost	CPU	Cost	CPU	GAP	Cost	CPU	GAP	Cost	CPU	GAP
20-5-1a	54,793	25	54,793	18	0.00	54,793	12	0.00	54,793	28	0.00
20-5-1b	39,104	22	39,104	13	0.00	39,104	10	0.00	39,104	32	0.00
20-5-2a	48,908	33	48,908	15	0.00	48,908	8	0.00	48,908	41	0.00
20-5-2b	37,542	38	37,542	19	0.00	37,542	11	0.00	37,542	47	0.00
50-5-1a	90,111	39	91,849	25	1.93	91,646	20	1.70	91,172	42	1.18
50-5-1b	63,242	40	63,242	31	0.00	63,242	19	0.00	63,242	54	0.00
50-5-2a	88,298	46	88,838	41	0.61	88,786	22	0.55	88,298	57	0.00
50-5-2b	67,308	37	67,449	23	0.21	67,484	17	0.26	67,340	43	0.05
50-5-2bis	84,055	33	84,172	27	0.14	84,110	20	0.07	84,055	39	0.00
50-5-2Bbis	51,822	48	52,059	33	0.46	52,059	29	0.46	51,843	37	0.04
50-5-3a	86,203	54	89,914	49	4.30	86,757	25	0.64	86,203	60	0.00
50-5-3b	61,830	31	61,885	28	0.09	61,830	16	0.00	61,830	44	0.00
100-5-1a	275,066	245	283,802	212	3.18	282,357	170	2.65	281,975	262	2.51
100-5-1b	213,618	164	216,295	155	1.25	214,484	134	0.41	214,446	199	0.39
100-5-2a	193,671	176	195,156	142	0.77	195,120	119	0.75	195,303	184	0.84
100-5-2b	157,129	114	157,636	134	0.32	157,178	95	0.03	157,286	195	0.10
100-5-3a	200,079	182	201,474	170	0.70	200,851	121	0.39	201,407	234	0.66
100-5-3b	152,441	282	155,022	315	1.69	154,429	171	1.30	154,119	334	1.10
100-10-1a	288,843	166	295,552	101	2.32	294,845	131	2.08	293,371	202	1.57
100-10-1b	231,028	243	236,674	149	2.44	234,526	156	1.51	233,527	321	1.08
100-10-2a	243,590	167	246,586	130	1.23	245,099	140	0.62	244,429	184	0.34
100-10-2b	203,988	183	206,379	112	1.17	206,158	144	1.06	205,830	234	0.90
100-10-3a	250,882	223	255,480	149	1.83	254,064	194	1.27	253,547	315	1.06
100-10-3b	204,188	191	206,809	134	1.28	206,759	188	1.26	206,759	187	1.26
200-10-1a	476,516	828	489,834	954	2.79	488,841	794	2.59	486,285	919	2.05
200-10-1b	377,809	974	389,516	960	3.10	389,857	766	3.19	389,143	1002	3.00
200-10-2a	449,470	1120	460,655	1050	2.49	458,991	889	2.12	455,810	1212	1.41
200-10-2b	375,159	1092	380,581	1132	1.45	380,997	781	1.56	380,049	1194	1.30
200-10-3a	472,962	891	488,191	940	3.22	486,431	662	2.85	485,058	955	2.56
200-10-3b	364,737	1276	376,791	1011	3.30	374,593	941	2.70	372,048	1194	2.00
Ave.		299		276	1.41		227	1.07		328	0.85

TABLE IV: BARRETO'S DATA SET, COMPARISON WITH BKS

problem ID	n	m	Q	BKS	Cost		CPU		GAP		C. V
					1 run	10 run	1 run	10 run	1 run	10 run	
Christofides69-50x5	50	5	160	565.6	565.6	565.6	21	16	0.00	0.00	0.000
Christofides69-75x10	75	10	140	844.4	854.03	848.9	90	101	1.14	0.53	0.002
Christofides69-100x10	75	10	200	833.4	847.1	837.14	178	160	1.64	0.45	0.005
Daskin95-88x8	88	8	9000000	355.8	356.03	355.8	109	82	0.06	0.00	0.013
Daskin95-150x10	150	10	8000000	43919.9	44660.3	44159	355	342	1.69	0.54	0.029
Gaskell67-21x5	21	5	6000	424.9	424.9	424.9	8	9	0.00	0.00	0.000
Gaskell67-22x5	22	5	4500	585.1	585.1	585.1	9	9	0.00	0.00	0.000
Gaskell67-29x5	29	5	4500	512.1	512.1	512.1	12	12	0.00	0.00	0.000
Gaskell67-32x5	32	5	8000	562.2	562.2	562.2	9	11	0.00	0.00	0.000
Gaskell67-32x5	32	5	11000	504.3	504.3	504.3	21	16	0.00	0.00	0.000
Gaskell67-36x5	36	5	250	460.4	460.4	460.4	16	17	0.00	0.00	0.000
Min92-27x5	27	5	2500	3062	3062	3062	9	13	0.00	0.00	0.000
Min92-134x8	134	8	850	5709	5853.51	5716.42	211	231	2.53	0.13	0.013
Ave.							81	78	0.54	0.13	0.005

B. Barreto's Instances

The first set is given by Barreto et al. [3], containing 13 instances available at <http://sweet.ua.pt/~sbarreto/>. Their traveling cost is assumed to be the Euclidean distances (not rounded), and all routes are capacitated. An interesting feature of this data set is the existence of both lower and upper bounds provided in Barreto et al. [3]. These instances are either from the literature or are modified by adding both capacitated and incapacitated depots to the classical VRP instances. The number of candidate sites varies from 5 to 10 while the numbers of customers vary from 21 to 150. The vehicle variable cost is not considered in this set of instances.

Table IV summarizes the computational results of the proposed algorithm associated with Barreto's data set. Taking Table IV into account, the average gaps for single and multiple runs are 0.54, and 0.13, respectively. This proposed method reaches the best existence results in 9 out of 13 instances and for the rest of the instances has relatively good performance with a maximum gap of 0.54%. Multiple runs lead to a decrease in the average gap from 0.54 to 0.13 and also increases the number of best obtained solutions from 8 to 9. Taking into consideration Table IV it can be concluded that the proposed algorithm with multiple runs outperforms all other methods in terms of solution quality with the average gap of 0.13 except GRASP×ELS, and MACO with average gap quality of 0.08 and 0.07, respectively.

C. Tuzun and Burke's Instances

The third set of instances was first described in Tuzun and Burke [22], that includes 36 instances. The instances are available at <http://prod.honc.free.fr/>. Different from other two sets of instances, there is no capacity restrictions for facilities. Number of customers are chosen from the set $n \in \{100, 150, 200\}$, and number of depots are either 10 or 20, vehicle have homogenous capacity ($Q = 150$) and demands are generated uniformly in interval $[1, 20]$. Traveling costs corresponds to the Euclidean distances and are not rounded.

Table V denotes that the average gap for single and multiple runs are 1.12, and 0.68, respectively. The number

of best-known solutions is 2 and 6 for single and multiple runs respectively. In addition, the solution of instance 13-31-22 is improved ($GAP = -0.04$).

According to Table VI, which is presented for comparison with other algorithms, it is quite evident that HVNS with multiple runs outperforms strictly all other selected heuristic approaches both in terms of average gap as well as the number of best solutions. The performance of single run HVNS is not as good as multiple runs such that the average gaps increase from 0.13 to 1.12. The multiple runs of HVNS reaches 10 best solutions among the 7 tested methods.

TABLE V: SUMMARIZES THE COMPUTATIONAL RESULTS OF THE PROPOSED ALGORITHM ASSOCIATED WITH THIS DATA SET

Problem ID	GRASP		MA/PM		LRGTS		SALRP		GRASP*ELS		2-Phase HGTS		MACO		HVNS	
	GAP	CPU	GAP	CPU	GAP	CPU	GAP	CPU	GAP	CPU	GAP	CPU	GAP	CPU	GAP	CPU
Christofides69-50x5	5.92	2	0.00	0	3.68	2	0.00	53	0.00	8	2.62	45	0.00	29	0.00	16
Christofides69-75x10	2.04	10	2.57	1	2.26	10	0.43	127	0.76	86	0.53	94	0.06	59	0.53	101
Christofides69-100x10	3.38	26	2.00	1	1.14	28	0.59	331	0.00	127	0.62	234	0.40	84	0.45	160
Daskin95-88x8	0.31	17	0.00	0	3.63	18	0.00	227	0.00	130	1.74	148	0.00	100	0.00	82
Daskin95-150x10	1.61	156	0.21	0	1.06	119	2.71	577	0.10	1697	1.50	456	0.48	167	0.54	342
Gaskell67-21x5	1.11	0	0.00	0	0.00	0	0.00	18	0.00	0	0.00	6	0.00	6	0.00	9
Gaskell67-22x5	0.00	0	4.56	5	0.39	0	0.00	17	0.00	15	0.00	9	0.00	5	0.00	9
Gaskell67-29x5	0.59	0	0.00	0	0.00	0	0.00	24	0.00	9	0.00	11	0.00	9	0.00	12
Gaskell67-32x5	1.73	1	1.73	2	3.98	1	0.00	27	0.00	18	0.00	40	0.00	13	0.00	11
Gaskell67-32x5	0.00	1	6.03	6	0.10	1	0.00	25	0.00	34	0.00	22	0.00	10	0.00	16
Gaskell67-36x5	0.00	1	5.43	5	3.50	1	0.00	32	0.00	0	0.00	39	0.00	13	0.00	17
Min92-27x5	0.00	0	0.00	0	0.10	0	0.00	23	0.00	35	0.00	11	0.00	9	0.00	13
Min92-134x8	4.49	50	4.22	3	1.75	48	0.00	522	0.18	280	3.18	252	0.00	137	0.13	231
Global Ave.	1.63	20	2.06	2	1.66	18	0.29	154	0.08	188	0.78	105	0.07	49	0.13	78
Num. of Best	4		5		2		10		11		7		11		9	

TABLE VI: TUZUN AND BURKE'S DATA SET, COMPARISON WITH BKS

problem ID	n	m	Q	BKS	Cost		CPU		GAP		C. V
					1 run	10 run	1 run	10 run	1 run	10 run	
11-11-12	100	10	150	1467.68	1474.28	1473.20	98	114	0.45	0.38	0.000
11-11-22	100	20	150	1449.20	1460.80	1449.20	152	147	0.80	0.00	0.015
11-12-12	100	10	150	1394.80	1395.87	1394.80	148	140	0.08	0.00	0.013
11-12-22	100	20	150	1432.29	1433.87	1432.29	112	146	0.11	0.00	0.009
11-21-12	100	10	150	1167.16	1174.62	1167.54	132	116	0.64	0.03	0.003
11-21-22	100	20	150	1102.24	1102.24	1102.24	149	178	0.00	0.00	0.000
11-22-12	100	10	150	791.66	793.25	792.03	153	137	0.20	0.05	0.005
11-22-22	100	20	150	728.30	728.30	728.30	103	128	0.00	0.00	0.004
11-31-12	100	10	150	1238.24	1245.33	1240.08	111	101	0.57	0.15	0.005
11-31-22	100	20	150	1245.31	1248.16	1245.54	138	142	0.23	0.02	0.001
11-32-12	100	10	150	902.26	903.30	902.34	191	201	0.12	0.01	0.002
11-32-22	100	20	150	1018.29	1020.14	1019.10	127	115	0.18	0.08	0.002
13-11-12	150	10	150	1866.75	1954.20	1942.67	283	241	4.68	4.07	0.017
13-11-22	150	20	150	1833.95	1850.50	1850.50	157	193	0.90	0.90	0.010
13-12-12	150	10	150	1965.12	2005.73	2000.03	208	227	2.07	1.78	0.012
13-12-22	150	20	150	1801.39	1819.89	1819.89	276	311	1.03	1.03	0.019
13-21-12	150	10	150	1443.33	1467.94	1461.71	261	204	1.71	1.27	0.016
13-21-22	150	20	150	1441.98	1500.44	1446.89	202	252	4.05	0.34	0.023
13-22-12	150	10	150	1204.42	1209.67	1208.50	294	262	0.44	0.34	0.009
13-22-22	150	20	150	930.99	934.81	932.38	234	243	0.41	0.15	0.002
13-31-12	150	10	150	1699.22	1728.41	1716.22	318	306	1.72	1.00	0.015
13-31-22	150	20	150	1400.01	1401.81	1399.42	288	297	0.13	-0.04	0.007
13-32-12	150	10	150	1199.51	1204.22	1204.22	184	216	0.39	0.39	0.008
13-32-22	150	20	150	1152.18	1157.64	1156.32	231	202	0.47	0.36	0.002
12-11-12	200	10	150	2259.87	2322.02	2286.76	537	614	2.75	1.19	0.012
12-11-22	200	20	150	2166.43	2216.44	2216.44	707	801	2.31	2.31	0.014
12-12-12	200	10	150	2231.46	2260.53	2260.53	849	702	1.30	1.30	0.019
12-12-22	200	20	150	2237.81	2310.86	2264.63	694	645	3.26	1.20	0.017
12-21-12	200	10	150	2089.77	2111.10	2102.32	548	631	1.02	0.60	0.006
12-21-22	200	20	150	1709.56	1730.14	1717.79	738	712	1.20	0.48	0.024
12-22-12	200	10	150	1466.62	1475.92	1470.40	653	637	0.63	0.26	0.008
12-22-22	200	20	150	1082.59	1093.14	1090.98	694	720	0.97	0.77	0.005
12-31-12	200	10	150	1970.44	1990.86	1987.82	747	779	1.04	0.88	0.005
12-31-22	200	20	150	1918.93	1971.43	1957.07	884	864	2.74	1.99	0.001
12-32-12	200	10	150	1771.06	1789.33	1780.13	519	631	1.03	0.51	0.024
12-32-22	200	20	150	1390.87	1400.84	1400.84	1011	918	0.72	0.72	0.310
Ave.							365	369	1.12	0.68	0.018

TABLE VII: TUZUN AND BURKE'S DATA SET, COMPARISON WITH OTHER ALGORITHMS

Problem ID	GRASP		MAIPM		LRGTS		SALRP		GRASP*ELS		2-Phase HGTS		MACO		HVNS	
	GAP	CPU	GAP	CPU	GAP	CPU	GAP	CPU	GAP	CPU	GAP	CPU	GAP	CPU	GAP	CPU
11-11-12	3.92	32	1.79	32	1.58	3	0.65	369	0.39	233	0.79	152	1.50	71	0.38	114
11-11-22	5.36	41	1.53	36	1.56	7	1.50	274	0.00	9	2.56	239	0.32	46	0.00	147
11-12-12	2.06	28	1.72	36	1.24	4	0.99	231	0.13	112	0.89	120	0.93	61	0.00	140
11-12-22	3.49	36	4.20	36	0.75	7	0.00	420	0.00	114	2.91	146	0.08	54	0.00	146
11-21-12	2.83	28	0.52	32	1.75	7	0.86	348	0.00	27	0.00	232	3.50	80	0.03	116
11-21-22	1.94	34	1.19	43	1.24	7	0.74	342	0.00	259	0.00	224	0.00	65	0.00	178
11-22-12	2.82	23	0.29	38	2.73	5	0.00	360	0.05	5	0.00	201	0.16	95	0.05	137
11-22-22	2.68	37	0.30	49	2.01	6	0.50	418	0.00	48	0.00	254	0.00	65	0.00	128
11-31-12	2.82	22	1.94	37	2.40	4	0.02	300	0.17	55	0.02	160	2.18	77	0.15	101
11-31-22	2.22	36	0.48	48	0.87	6	0.16	428	0.06	233	0.47	237	0.93	50	0.02	142
11-32-12	1.10	20	0.17	35	1.20	4	0.00	291	0.00	249	0.00	135	0.00	61	0.01	201
11-32-22	0.41	38	0.46	63	0.71	5	0.56	316	0.00	196	0.00	157	0.00	69	0.08	115
Ave.	2.64	31	1.22	40	1.50	5	0.50	341	0.07	128	0.64	188	0.80	66	0.06	139
13-11-12	7.50	113	4.96	129	4.25	13	4.67	743	4.17	518	5.09	485	4.21	227	4.07	241
13-11-22	3.00	161	2.60	144	2.28	19	3.55	835	1.65	705	1.23	298	1.05	101	0.90	193
13-12-12	3.50	100	0.97	111	2.31	11	4.70	456	1.39	727	2.42	406	1.34	201	1.78	227
13-12-22	3.04	132	2.99	144	1.03	16	0.00	833	1.88	415	0.09	302	0.61	141	1.03	311
13-21-12	4.50	118	0.34	167	0.37	22	0.69	750	0.72	103	0.13	449	3.86	206	1.27	204
13-21-22	1.03	166	1.24	155	3.53	28	0.94	828	0.15	662	0.70	493	0.32	163	0.34	252
13-22-12	2.99	107	0.25	161	0.55	15	0.15	752	1.28	459	0.00	270	0.03	218	0.34	262
13-22-22	1.05	142	0.41	196	0.64	14	0.39	842	1.59	224	0.05	335	0.08	150	0.15	243
13-31-12	2.22	93	1.24	144	1.77	18	1.27	742	0.76	271	0.36	444	1.46	226	1.00	306
13-31-22	1.84	128	2.09	156	1.76	19	1.13	833	0.21	524	1.19	342	0.07	123	-0.04	297
13-32-12	2.02	89	0.33	154	1.40	15	1.44	756	1.28	251	2.94	526	1.48	241	0.39	216
13-32-22	6.87	135	0.55	223	0.87	14	0.60	837	0.33	375	0.34	380	0.51	130	0.36	202
Ave.	3.30	124	1.50	157	1.73	17	1.63	767	1.28	436	1.21	394	1.25	177	0.96	246
12-11-12	5.49	308	1.51	418	1.62	33	2.84	1328	1.59	655	0.25	522	1.98	461	1.19	614
12-11-22	5.62	410	5.12	458	1.90	40	4.23	1455	1.71	432	0.00	603	0.98	231	2.31	801
12-12-12	1.87	311	1.93	377	1.32	33	1.29	1319	0.67	1566	0.80	527	0.00	428	1.30	702
12-12-22	4.79	419	6.19	436	0.97	40	3.96	1428	1.24	2192	0.00	558	1.69	234	1.20	645
12-21-12	2.26	338	0.79	351	1.48	47	1.09	1320	0.80	1521	1.54	522	0.42	569	0.60	631
12-21-22	5.72	370	3.62	377	1.65	59	0.79	1400	4.06	618	2.31	691	0.10	276	0.48	712
12-22-12	2.05	243	0.06	322	1.50	37	0.17	1299	0.52	514	0.45	724	0.43	544	0.26	637
12-22-22	1.23	309	0.50	505	0.74	39	0.56	1429	0.29	1243	0.00	616	0.46	316	0.77	720
12-31-12	3.77	283	0.14	413	0.69	42	1.20	1318	1.72	1451	0.73	542	0.42	386	0.88	779
12-31-22	8.96	399	3.13	406	3.52	52	0.68	1412	2.37	1273	2.09	617	2.13	230	1.99	864
12-32-12	1.00	199	0.63	353	0.89	34	0.45	1314	0.44	1398	0.42	697	0.67	405	0.51	631
12-32-22	1.28	296	0.39	530	0.74	43	0.40	1427	4.53	2202	0.00	518	0.13	268	0.72	918
Ave.	3.67	324	2.00	412	1.42	41	1.47	1371	1.66	1255	0.72	595	0.78	362	1.02	721
Global Ave.	3.20	160	1.57	203	1.55	21	1.20	826	1.00	607	0.86	392	0.95	202	0.68	369
Num. of Best	0		4		0		5		9		16		7		10	

IV. CONCLUSION

In this paper we proposed a hybrid variable neighborhood search (HVNS) to solve the capacitated location-routing problem (CLRP). The proposed algorithm takes advantage of simulated annealing criterion and a dynamic penalty function to explore the solution space more effectively. In addition, a new effective neighborhood search is also introduced. The computational experiments are carried out on three well-known sets of instances from the literature. The proposed HVNS is able to obtain optimal or near-optimal solutions in a reasonable computation time for most benchmark instances. It reaches 21 (single run) and 39 (multiple runs) best known solutions and updates 1 (single run) and 6 (multiple runs) best known solutions in 79 instances considered in this study. The obtained results demonstrate the effectiveness of the proposed method in providing promising results without extensive computational time.

REFERENCES

- [1] Afroz Ansari pour and Theodore B Trafalis. A robust multicriteria optimization model for city logistic terminal locations. In *IIE Annual Conference. Proceedings*, page 2801. Institute of Industrial and Systems Engineers (IISE), 2013.
- [2] Afroz Ansari pour, Adriana Mata, Sara Nourazari, Hillel Kumin, et al. Some explicit results for the distribution problem of stochastic linear programming. *Open Journal of Optimization*, 5(04):140, 2016.
- [3] Sergio Barreto, Carlos Ferreira, Jose Paixao, and Beatriz Sousa Santos. Using clustering analysis in a capacitated location-routing problem. *European Journal of Operational Research*, 179(3): 968–977, 2007.
- [4] Christophe Duhamel, Philippe Lacomme, Christian Prins, and Caroline Prodhon. A grasp els approach for the capacitated location-routing problem. *Computers & Operations Research*, 37(11): 1912–1923, 2010.
- [5] John Willmer Escobar, Rodrigo Linfati, and Paolo Toth. A two-phase hybrid heuristic algorithm for the capacitated location-routing problem. *Computers & Operations Research*, 40(1):70–79, 2013.
- [6] Seyed Sajjad Fazeli, Saravanan Venkatachalam, Ratna Babu Chinnam, and Alper Murat. Two-stage stochastic choice modeling approach for electric vehicle charging station network design in urban communities. *IEEE Transactions on Intelligent Transportation Systems*, 2020.
- [7] Seyed Sajjad Fazeli, Saravanan Venkatachalam, and Jonathon M Smereka. Efficient algorithms for autonomous electric vehicles' min-max routing problem. *arXiv preprint arXiv:2008.03333*, 2020.

- [8] Pierre H Hansen, B Hegedahl, S Hjortkjaer, and Børge Obel. A heuristic solution to the warehouse location-routing problem. *European Journal of Operational Research*, 76(1):111–127, 1994.
- [9] Bassem Jarboui, Houda Derbel, Saïd Hanafi, and Nenad Mladenovic'. Variable neighborhood search for location routing. *Computers & Operations Research*, 40(1):47–57, 2013.
- [10] Rui Borges Lopes, Carlos Ferreira, Beatriz Sousa Santos, and Sérgio Barreto. A taxonomical analysis, current methods and objectives on location-routing problems. *International Transactions in Operational Research*, 20(6):795–822, 2013.
- [11] Yannis Marinakis and Magdalene Marinaki. A particle swarm optimization algorithm with path relinking for the location routing problem. *Journal of Mathematical Modelling and Algorithms*, 7(1):59–78, 2008.
- [12] Yannis Marinakis, Magdalene Marinaki, and Nikolaos Matsatsinis. Honey bees mating optimization for the location routing problem. In *Engineering Management Conference, 2008. IEMC Europe 2008. IEEE International*, pages 1–5. IEEE, 2008.
- [13] Nenad Mladenovic' and Pierre Hansen. Variable neighborhood search. *Computers & Operations Research*, 24(11):1097–1100, 1997.
- [14] Gábor Nagy and Saïd Salhi. Location-routing: Issues, models and methods. *European Journal of Operational Research*, 177(2):649–672, 2007.
- [15] Jossef Perl and Mark S Daskin. A warehouse location-routing problem. *Transportation Research Part B: Methodological*, 19(5):381–396, 1985.
- [16] Jossef Perl and Mark S Daskin. A warehouse location-routing problem. *Transportation Research Part B: Methodological*, 19(5):381–396, 1985.
- [17] Christian Prins, Caroline Prodhon, and Roberto Wolfler Calvo. A memetic algorithm with population management (ma| pm) for the capacitated location-routing problem. In *Evolutionary computation in combinatorial optimization*, pages 183–194. Springer, 2006.
- [18] Christian Prins, Caroline Prodhon, and Roberto Wolfler Calvo. Solving the capacitated location- routing problem by a grasp complemented by a learning process and a path relinking. *4OR*, 4(3): 221–238, 2006.
- [19] Christian Prins, Caroline Prodhon, Angel Ruiz, Patrick Soriano, and Roberto Wolfler Calvo. Solv- ing the capacitated location-routing problem by a cooperative lagrangean relaxation-granular tabu search heuristic. *Transportation Science*, 41(4):470–483, 2007.
- [20] Caroline Prodhon and Christian Prins. A survey of recent research on location-routing problems. *European Journal of Operational Research*, 2014.
- [21] Ching-Jung Ting and Chia-Ho Chen. A multiple ant colony optimization algorithm for the capacitated location routing problem. *International Journal of Production Economics*, 141(1):34–44, 2013.
- [22] Dilek Tuzun and Laura I Burke. A two-phase tabu search approach to the location routing problem. *European Journal of Operational Research*, 116(1):87–99, 1999.
- [23] Tai-Hsi Wu, Chinyao Low, and Jiunn-Wei Bai. Heuristic solutions to multi-depot location-routing problems. *Computers & Operations Research*, 29(10):1393–1415, 2002.
- [24] Vincent F Yu, Shih-Wei Lin, Wenyih Lee, and Ching-Jung Ting. A simulated annealing heuristic for the capacitated location routing problem. *Computers & Industrial Engineering*, 58(2):288–299, 2010.