Combining Biased Randomization with Iterated Local Search for Solving the Multi-Depot Vehicle Routing Problem

ARTICLE *in* INTERNATIONAL TRANSACTIONS IN OPERATIONAL RESEARCH · MAY 2014

Impact Factor: 0.98 · DOI: 10.1111/itor.12101

4 AUTHORS, INCLUDING:

Angel A. Juan

Universitat Oberta de Catalunya

139 PUBLICATIONS **478** CITATIONS

SEE PROFILE

Daniel Guimarans

Hogeschool van Amsterdam

36 PUBLICATIONS **26** CITATIONS

SEE PROFILE

Barry Barrios

14 PUBLICATIONS **104** CITATIONS

SEE PROFILE

COMBINING BIASED RANDOMIZATION WITH ITERATED LOCAL SEARCH FOR SOLVING THE MULTI-DEPOT VEHICLE ROUTING PROBLEM

Angel A. Juan[1], Iñaki Pascual[1], Daniel Guimarans[2], Barry Barrios[1]

(1) IN3-Computer Science Dept., Open University of Catalonia, Barcelona, Spain

(2) National ICT Australia, Sydney, Australia

ABSTRACT

This paper proposes a hybrid approach for solving the Multi-Depot Vehicle Routing Problem (MDVRP) with a limited number of identical vehicles per depot. Our approach, which only uses a few parameters, combines 'biased randomization' –use of non-symmetric probability distributions to generate randomness– with the Iterated Local Search (ILS) metaheuristic. Two biased-randomized processes are employed at different stages of the ILS framework in order to: (a) assign customers to depots following a randomized priority criterion –this allows for fast generation of alternative allocation maps; and (b) improving routing solutions associated with a 'promising' allocation map –this is done by randomizing the classical savings heuristic. These biased-randomized processes rely on the use of the geometric probability distribution, which is characterized by a single and bounded parameter. Being an approach with few parameters, our algorithm does not require troublesome fine-tuning processes, which tend to be time consuming. Using standard benchmarks, the computational experiments show the efficiency of the proposed algorithm. Despite its hybrid nature, our approach is relatively easy to implement and can be parallelized in a very natural way, which makes it an interesting alternative for practical applications of the MDVRP.

Keywords: Multi-Depot Vehicle Routing Problem, Metaheuristics, Randomized Algorithms, Biased Randomization, Iterated Local Search, Parallelization of Heuristics.

1. INTRODUCTION

The Capacitated Vehicle Routing Problem (CVRP) is probably the most popular routing problem in the literature on combinatorial optimization. The basic goal is to find an optimal set of routes for a fleet of vehicles so that a set of customers' demands is satisfied. Usually all vehicles are considered to be identical (homogeneous fleet). All routes begin and end at one depot, where all resources are initially located. Typically each vehicle has a maximum loading capacity, a single vehicle supplies each customer, and a vehicle cannot stop twice at the same

customer. The classical objective is to minimize the costs related to distances travelled by vehicles and/or the times spent during the distribution process, while satisfying the associated constraints.

As a generalization of the Travelling Salesman Problem, the CVRP which has been studied for decades, is NP-hard **(Prins, 2004; Laporte, 2007)**. Nevertheless, it is still attracting a great amount of attention from researchers due to its potential applications, both to real-life scenarios and to the development of new algorithms, optimization methods, and metaheuristics for solving other combinatorial problems **(Toth and Vigo, 2002; Golden et al., 2008)**. Different approaches to the CVRP have been explored, ranging from the use of pure optimization methods, such as mixed-integer linear programming –mainly used for solving small and medium-size problems–, to the use of heuristics and metaheuristics that provide near-optimal solutions for medium- and large-size problems in reasonable computing times. One CVRP variant which usually can be found in many real-life scenarios is the so-called Multi-Depot Vehicle Routing Problem (MDVRP). This is a challenging problem since it integrates a combinatorial assignment problem –which customers are to be assigned to each depot– with the several CVRPs that must be solved for each customers-depot allocation map **(Figure 1)**. In an MDVRP, allocation and routing issues are often interrelated. Thus, it is not trivial to find the customers-depots allocation map that provides the optimal routing solution for the entire set of customers and depots.

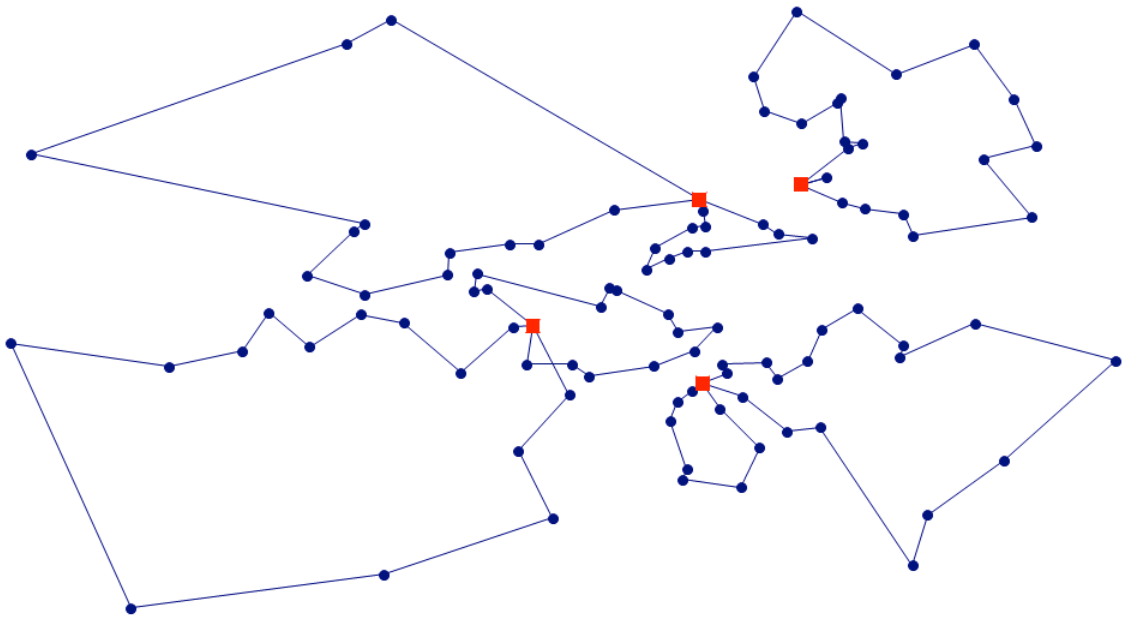


Figure 1: A medium-size MDVRP with 96 customers (circles) and 4 depots (squares).

The MDVRP may be formally described as an extension of the CVRP. It is defined as a complete undirected graph $G = \{V, E\}$, where $V = \{V_d, V_c\}$ is the set of nodes including the

depots (V_d) and the customers (V_c), and E is the set of edges connecting all nodes in V. Each customer in V_c has a positive demand to be satisfied. Each edge in E has an associated cost $c_{ij} = c_{ji} > 0$, usually computed as the distance between customers i and j (all distances are assumed to be symmetric); c_{ik} is the distance-based cost between the customer $i \in V_c$ and the depot $k \in V_d$. For the MDVRP, a solution is a set of routes in which each route starts at one depot in V_d, connects one or more customers in V_c, and ends at the same depot, without exceeding the capacity of the vehicle. Also routes that start at one depot and end at another depot are not allowed.

The number of vehicles based at each depot (m) may be fixed or unlimited. The former defines a harder problem, since it adds an additional constraint and there is also no guarantee that a feasible solution exists **(Chao et al., 1993)**. The latter simplifies the modelling and solving of the MDVRP. However, in a realistic scenario, the number of available vehicles to satisfy customers' demands is usually limited. As it will be shown in the experimental section, our approach is able to efficiently deal with both cases, either with or without a limitation in the number of vehicles per depot.

This paper proposes a hybrid approach that combines an ILS-like metaheuristic **(Lourenço et al., 2003, 2010)** with biased-randomization techniques **(Juan et al., 2009, 2011b)** to deal with the MDVRP. This hybrid framework has been recently used to develop efficient algorithms in both scheduling and routing problems **(Dominguez et al., 2014; Juan et al., 2014)**. By randomizing some steps in a deterministic heuristic, it is transformed into a probabilistic procedure. Then, it can be run multiple times –either in sequential or parallel mode– in order to obtain different outcomes or solutions. Moreover, it is possible to perform this randomization process without losing the logic behind the deterministic heuristic by employing an asymmetric (biased) probability distribution. Therefore, alternative solutions of similar quality can be easily generated. Being an algorithm with few parameters, our approach represents an interesting alternative to other state-of-the-art metaheuristics, which usually require more cumbersome and time-costly fine-tuning processes. In practice, these approaches are usually harder to implement and often not reproducible. To the best of our knowledge, it is the first time that biased-randomization techniques are used to efficiently solve the multi-depot version of the VRP. In effect, while most existing metaheuristics for the MDVRP are heavily based on local search, the proposed method –similarly to GRASP and Ant Colony Optimization approaches–, relies on biased constructive procedures to generate new solutions. This class of methods has been less studied in the MDVRP literature and the paper shows they can offer competitive results when compared to other approaches known from the literature that require more complex and time-consuming parameter fine-tuning processes.

The rest of this article is structured as follows. The next section reviews the literature published on the MDVRP and the different approaches adopted to solve it. Next, the biased randomization issues directly related to our approach are discussed. Afterwards, the main characteristics of our algorithm are outlined and explained, and some implementation details are given. Then, we present and analyze the results obtained from the application of our approach to some benchmark instances. Some parallelization issues are also considered and analyzed. Finally, the conclusion section summarizes the main contributions of our work.

2. LITERATURE REVIEW

The literature for the MDVRP dates back over 40 years. In the 1970s and 1980s, several heuristics for the MDVRP were proposed. For instance, **Tillman and Cain (1972)** develop an approach based on the well-known Clarke and Wright Savings (CWS) procedure **(Clarke and Wright, 1964)**. Thus, in **Wren and Holliday (1972)** the authors define a two-step approach able to solve instances with two depots. In **Gillett and Johnson (1976)** a three-step multi-terminal sweep algorithm is developed and applied to solve eleven benchmark instances. **Golden et al. (1977)** solve the MDVRP using two different heuristics: an improved version of the aforementioned Tillman and Cain's savings procedure, and a two-step extension of this approach intended for larger problems. **Raft (1982)** develops a three-step heuristic, which is used to solve four MDVRP benchmark instances. Also, **Laporte et al. (1984)** adopt an exact approach to the problem and propose a branch-and-bound algorithm that is used to solve 31 randomly generated test instances of up to 25 nodes including depots and customers.

Recently developed metaheuristics for the MDVRP were proposed, which includes Tabu Search, Genetic Algorithm, Adaptive Large Neighborhood Search, among others. In **Chao et al. (1993)**, the authors developed a record-to-record improvement heuristic for the MDVRP. **Renaud et al. (1996)** and **Cordeau et al. (1997)** both proposed a Tabu Search (TS) metaheuristics for solving the MDVRP. In particular, the TS algorithm developed by the former authors was tested on 23 benchmark instances with up to 360 nodes, obtaining the best-known solutions for all of them after completing a fine-tuning process. These authors also proposed a simplified version of their algorithm, which was able to obtain the best-known solutions for most instances in a reasonable time. **Cordeau et al. (1997)** obtained even better solutions than those obtained by **Renaud et al. (1996)**. **Thangiah and Salhi (2001)** developed a clustering method based on a Genetic Algorithm aimed to solve the MDVRP. **Chan et al. (2001)** addressed an MDVRP with both location and routing decisions and a high degree of uncertainty in the demands. In their work they were able to solve a large-scale stochastic location-routing problem. **Pisinger and Ropke (2007)** proposed an Adaptive Large

Neighborhood Search (ALNS) algorithm for solving the MDVRP and other VRPs. This method was tested on 33 benchmark instances, improving the best-known solution in 15 cases. However, the approach uses up to 14 parameters that are fine-tuned on a benchmark set containing 16 problems. **Crevier et al. (2007)** addressed an extension of the problem where vehicles may be replenished at intermediate depots along their routes. **Ho et al. (2008)** developed two Hybrid Genetic Algorithms (HGAs). In their approaches, these authors combine three different heuristics: the CWS, the Nearest Neighbor, and the Iterated Swap. The former two heuristics are used to generate initial solutions, whereas the last one is used to improve the solutions, including parents and offspring. A computational study was carried out to compare both HGAs. The HGA1, which generates initial solutions randomly, was proven to be inferior to HGA2, which applied the aforementioned heuristics to generate initial solutions. **Vidal et al. (2012)** also proposed a HGA to deal with different variants of the VRP, including the MDVRP. Using this approach, the authors identified the best-known solutions, including the optimal ones, or some new best solutions for all 33 benchmark instances described either in **Cordeau et al. (1997)** or in **Pisinger and Ropke (2007)**. Nevertheless, the HGA requires an in-deep adjustment of its components and the corresponding parameters. Also, **Mirabi et al. (2010)** propose three probabilistic hybrid heuristics, combining elements from both constructive heuristic search and improvement techniques, to cope with the MDVRP. Experiments were run on a number of randomly generated test problems of varying depots and up to 200 customers. **Dondo and Cerdá (2007)** present a novel three-phase approach for the Multi-Depot VRP with Time Windows and Heterogeneous vehicles (MDHVRPTW). This approach has been also used to provide an initial solution for a Large Scale Neighborhood (LSN) search method **(Dondo and Cerdá, 2009)**. **Ceselli et al. (2009)** and **Bettinelli et al. (2011)** consider the MDHVRPTW, too. The former introduces an exact column generation algorithm, while the latter proposes a branch-and-cut-and-price algorithm able to solve instances with up to 72 customers to proven optimality. Finally, **Cordeau and Maischberger (2012)** introduce an iterated TS heuristic to solve four different routing problems (the classical VRP, the periodic VRP, the multi-depot VRP, and the site-dependent VRP), combining a simple perturbation mechanism and TS as the local search procedure within an ILS framework. The perturbation mechanism is inspired from the cluster removal heuristic used in the ALNS **(Pisinger and Ropke, 2007)**. They also describe a parallel implementation of the heuristic to take advantage of multiple-core processors.

3. RANDOMIZATION ISSUES IN OUR APPROACH

In the approach presented in this paper, we use biased-randomization techniques at several stages of the ILS framework. As it is shown in this work, integrating these techniques inside a metaheuristic framework provides an efficient mechanism to solve challenging combinatorial optimization problems like the MDVRP.

Roughly speaking, Monte Carlo simulation or random sampling can be defined as a set of techniques that make use of random numbers and probability distributions to solve certain stochastic and deterministic problems. When properly combined with heuristic techniques, random sampling has proved to be extremely useful for solving stochastic VRPs **(Juan et al., 2011a, 2013)**. Likewise, approaches combining biased-randomization with heuristics can be used for solving efficiently deterministic routing problems, such as the CVRP. In particular, the SR-GCWS-CS algorithm **(Juan et al., 2009, 2011b)** introduces the geometric distribution to induce a biased-randomized process into the aforementioned CWS heuristic. With this mechanism, a new feasible and potentially good solution is generated every time the biased-randomized version of the CWS heuristic is executed. Thus, a set of different feasible solutions may be obtained by iterating this fast constructive method. By construction, each one of these feasible solutions consists of a set of round-trip routes from the depot that satisfy all problem constraints and node demands. This strategy can also be used to generate initial solutions for other metaheuristic approaches **(Guimarans et al., 2011; Juan et al., 2014)**.

The key of this process is how to randomize a given heuristic, the CWS in this case, so that the resulting solutions are still competitive in terms of associated costs. As in many other heuristics, at each step of the solution construction process, the CWS procedure chooses the first element of a list that has been previously sorted according to some logical criterion. In the particular case of the CWS heuristic, the first element in the list corresponds to the edge with the highest savings value. To partially avoid this greedy behavior without losing the general sense of the heuristic, the biased-randomized process assigns a selection probability to each element in the sorted list. In the case of the CWS, a different selection probability is assigned to each edge in the savings list. Of course, this probability should be coherent with the sorting criterion, e.g., the savings value associated with each edge. Thus, edges with a higher position in the list have a higher probability to be selected at the current iteration. Finally, this selection process is to be done without introducing too many parameters in the algorithm. Otherwise, it would require a fine-tuning process, which tends to be non-trivial, time-consuming, and often instance-dependent.

In order to reach the above-mentioned goals, biased probability functions, such as the single-parameter geometric distribution or the parameter-free decreasing triangular distribution, may be used during the randomized selection process. Therefore, if a geometric distribution is

applied, elements (edges in this case) with a higher position in the (savings) sorted list are more likely to be selected, but the exact probabilities depend on the specific value assigned to the distribution parameter p ($0 < p < 1$). **Figure 2** shows a comparison of two geometric probability distributions with different values of p. Their values are shown for the first elements in the sorted list of elements. It should be noticed that using relatively low values of p (e.g., $p = 0.1$) implies that more elements in the list are potentially eligible. On the other hand, using higher values for this parameter (e.g., $p = 0.4$) reduces the number of potentially eligible elements from the list. Finally, as the parameter p gets closer to 1, the greedy behavior of the heuristic is retrieved.

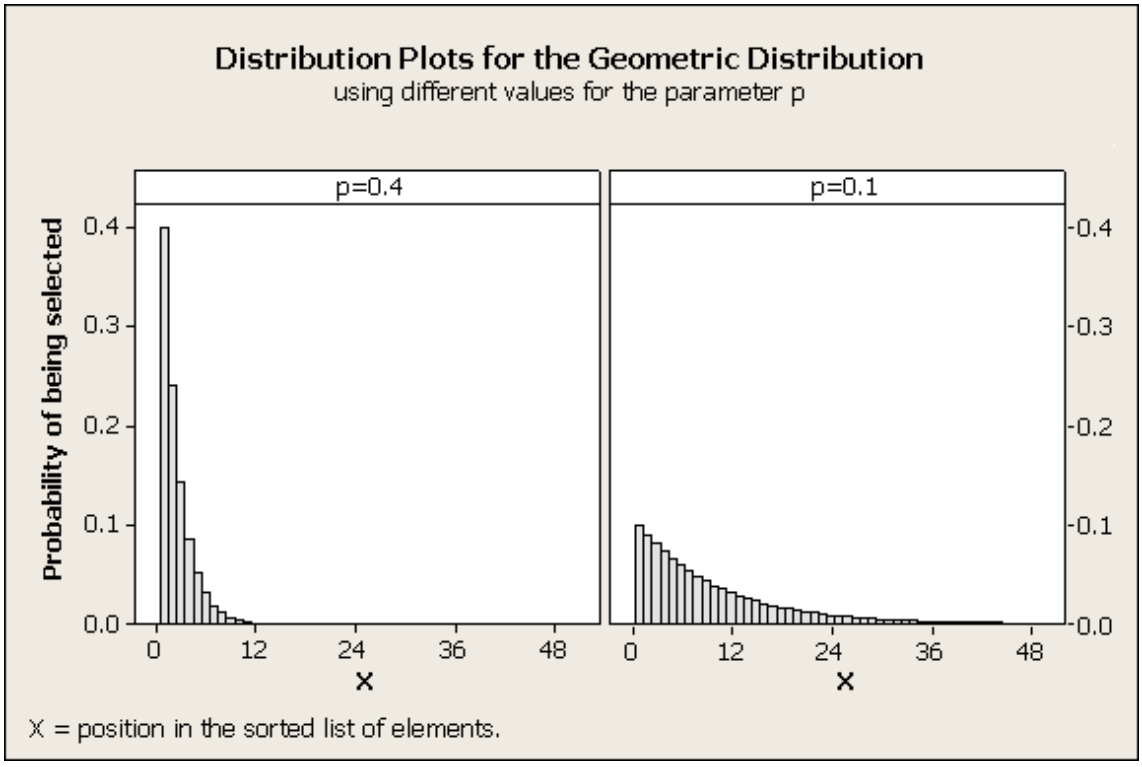


Figure 2: Geometric probability distributions using different values of the parameter p.

In this work, a similar process is also applied for randomizing the selection of customers serviced by each depot (customers-depots allocation maps). Nodes are assigned to depots according to distance- and capacity-related criteria, as explained later in the text. In both cases, all nodes are included in a sorted list, so at each step a depot selects the first element according to the applied criterion. Again, the geometric distribution is introduced to add some biased-randomized behavior to nodes selection. Therefore, different potentially good customers-depots allocation maps may be quickly generated by employing this simple mechanism, which uses a single parameter. Notice, however, that the value of the parameter p corresponding to this geometric distribution may be higher than the one employed to perform the randomization of

the CWS heuristic. This is due to the fact that the list of nodes to be assigned to a depot will be considerably shorter than the list of eligible edges in the CWS heuristic. At this point, it is important to notice that if a classical uniform randomization is used to drive the selection of customers by each depot, then any customer can be randomly assigned to any depot without following any distance-based priority strategy. Accordingly, the resulting customers-depots allocation maps are likely to be inferior in quality to the ones obtained using the proposed biased randomization approach –which uses a distance-based strategy as discussed next. In a series of experiments using common random numbers (to reduce the variance or random noise) and 1-minute computing time per instance, we have observed that the approach employing biased randomization –with the right parameter value– provides equal or better results in all 33 benchmark instances than the approach using uniform randomization, with an average gap being 1.82%.

4. AN OVERVIEW OF OUR APPROACH

The proposed approach to tackle the MDVRP is based on an Iterated Local Search framework **(Lourenço et al., 2010)** and the asymmetric randomization techniques described in the previous section. **Pseudo-code 1** shows how the biased randomization is integrated inside the traditional ILS framework. Basically, it is used to: *(a)* generate a random –but still efficient– initial solution, typically based on a classical heuristic; and *(b)* incorporate a 'common sense' (not uniform) policy or strategy to the randomness inside the perturbation process. A more detailed overview if provided by the flowchart diagram in **Figure 3**, which is described next.

```
procedure Iterated Local Search with Biased Randomization
  s0 = GenerateInitialSolution(biasedRand) % random but efficient
  s* = LocalSearch(s0)
  repeat
    s' = Perturbation(s*, history, biasedRand) % policy-oriented
    s*' = LocalSearch(s')
    s* = AcceptanceCriterion(s*, s*', history)
  until termination condition met
end
```

Pseudo-code 1: including biased randomization inside the ILS framework.

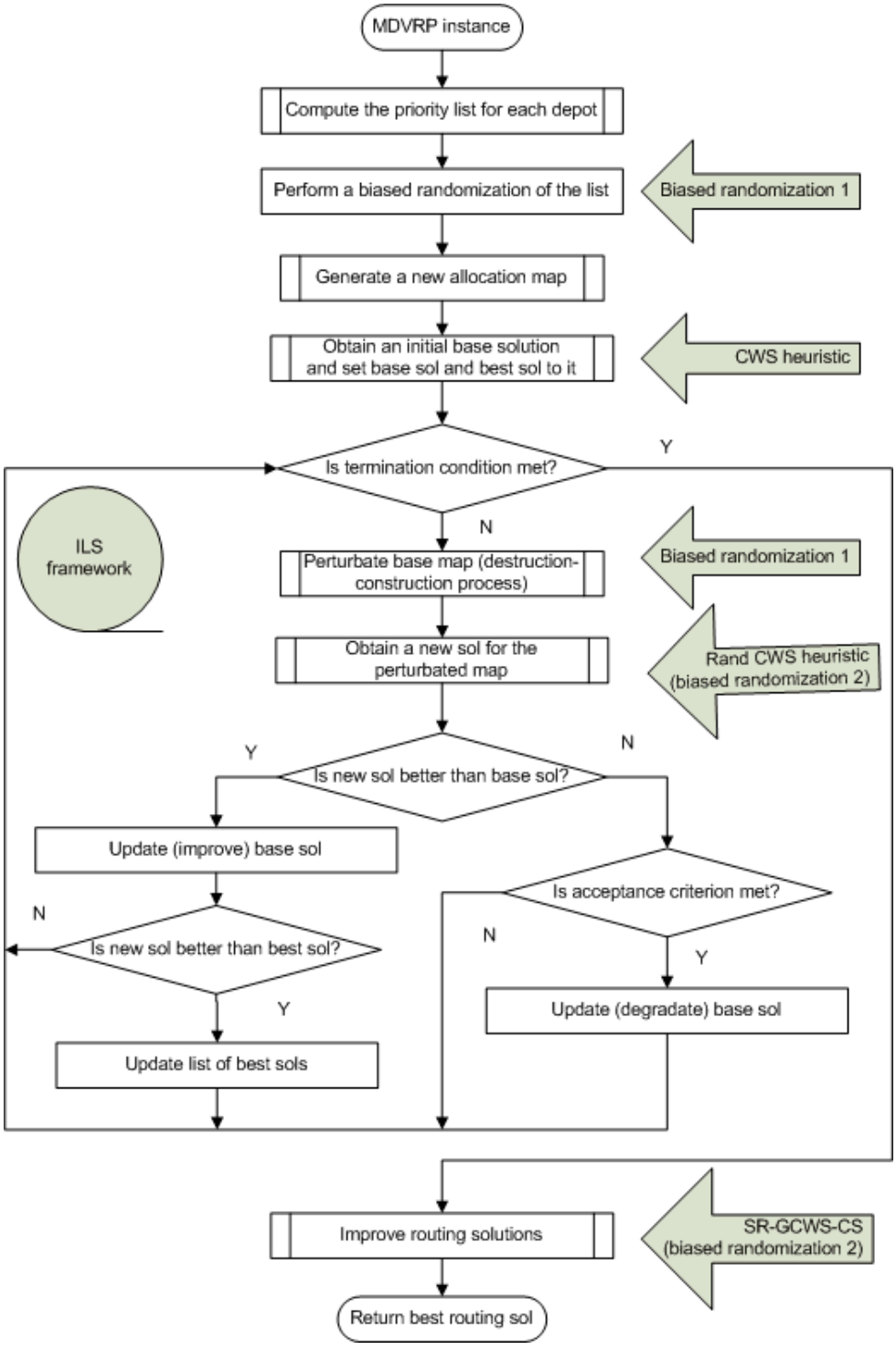


Figure 3: Flow diagram of the proposed approach.

First, a priority list of potentially eligible nodes is computed for each depot. Thus, $\forall k \in V_d$, the list associated with k contains all nodes, but they are sorted according to a distance-based criterion which depends on k. The distance-based criterion we use is called marginal savings,

and it is computed as follows $\forall i \in V_c$, where c_i^l represents the distance-based cost from the customer i to each depot l in V_d:

(a) We first compute the minimum cost of assigning customer i to k^*, the best alternative depot to k, i.e.: $c_i^{k^*} = \min_{\forall l \in V_d \setminus \{k\}} \{c_i^l\}$.

(b) Then, we compute μ_i^k, the marginal savings in cost associated with assigning customer i to depot k instead of assigning i to k^*: $\mu_i^k = c_i^{k^*} - c_i^k$.

Notice that these marginal savings can be either positive (if k happens to be the 'closest' depot to customer i) or negative (otherwise). According to this, nodes that are much closer to one depot k than to the other depots will present 'high' marginal-savings values for depot k and, accordingly will become a 'priority' for this depot –i.e. these nodes will occupy the top positions in the sorted priority list of depot k. On the contrary, nodes located in between two or more depots will present 'low' marginal-savings values for all depots, and they will not become a priority for any depot. The nodes list associated with each depot is sorted following this criterion. Then, the list is randomized through the geometric distribution as described in the previous section. The randomized list is used to assign nodes to depots. This can be done using different alternative criteria and, in fact, our method uses the following procedures at the starting stage as a way to generate initial nodes-depots assignment maps: *(i)* a round-robin tournament criterion following consecutive turns among depots is used to guarantee that a different depot selects a new node at a time –as far as it still has enough capacity to serve the associated demand; *(ii)* the same round robin criterion but this time a depot is randomly selected at each round for the node-selection turn; and *(iii)* at each round, the depot with the most (remaining) serving capacity selects the next node from its priority list –this criterion tends to generate quite 'balanced' allocation maps, i.e. customers-depots configurations in which the total demand to be served is similarly distributed among all depots. As described in the next section, for most instances it might be worthy to consider the different aforementioned criteria when generating the allocation maps. Once a depots-nodes allocation map is generated, a complete initial solution is obtained by individually solving each routing problem using the CWS heuristic, which is very fast and provides a reasonable good initial solution. Before entering the iterative process, the *base* and *best* solutions are updated with the values of the obtained initial solution.

At the next step, the ILS procedure is started. In this process, the base solution is perturbed by reallocating a certain percentage p^* of nodes in different depots. This way, a new allocation map is generated throughout a construction-destruction process. The reallocation of nodes into depots is guided according to the biased-randomized criteria introduced in the previous section.

The new routing problems associated with the new allocation map are solved again throughout the CWS heuristic, thus generating a new complete solution. If this new solution does not fulfill the limitation on the number of vehicles, then it is discarded and the random reallocation process is restarted. Otherwise, the base and best solutions are accordingly updated. Specifically, whenever the solution obtained is better than the current base solution, the base solution is updated. In addition, if the new solution is better than the best solution found so far, it is further improved by means of a fast local search process (details on this process are included in the next section). Afterwards, the base and best solutions are updated with the values of the new refined solution. Furthermore, in case the obtained solution is worse than the current base solution, an acceptance criterion is defined. This acceptance criterion allows a non-improving solution to be accepted as a new base solution if certain conditions are met; specifically, the algorithm allows downhill moves as far as the last movement in the previous iteration from x to x' was an improvement –i.e. $f(x') < f(x)$ for a minimization problem–, and the downhill move from x' to the new solution x'' is not greater than the last improvement –i.e., $|f(x'') - f(x')| < f(x) - f(x')$. This criterion is introduced in order to facilitate the method to escape from local minima and explore different regions of the search space.

During the ILS procedure, the top t (e.g. $t = 5$) best solutions found so far are saved as 'promising' solutions –notice that the routing of these solutions has been obtained using the fast CWS heuristic. Each of these top solutions is then further improved by means of a post-optimization process, which makes use of an efficient routing algorithm. In our case, the SR-GCWS-CS is employed **(Juan et al., 2011b).**

5. DETAILED PSEUDO-CODE

This section contains some technical details regarding our approach. In order to facilitate the actual implementation of the proposed methodology, we provide a pseudo-code version with a high level of details. The main procedure in our approach is shown in **Pseudo-code 2**. First, a priority list of potentially eligible nodes is calculated for each depot (lines 1 to 3) following the distance-savings criterion described before and detailed in **Pseudo-code 3**. At the second step of the algorithm, an initial solution is computed (lines 4 to 15). The priority list is biased-randomized using a geometric distribution (line 9). Next, a customers-depots allocation map is generated according to the randomized list of priorities (line 10). This process is detailed in **Pseudo-code 4**. Then, the corresponding routing problems are solved individually using the fast CWS routing heuristic (line 11). This process is outlined in **Pseudo-code 5**. Note that a feasibility checking might be needed in order to iterate the initial solution construction process until a feasible initial solution is obtained.

Regarding **Pseudo-code 4**, it should be noticed that different strategies or policies –previously described– are used in order to generate the depots-nodes allocation maps. As mentioned before, the last strategy is more conservative in the sense that it tends to provide more balanced maps. Balanced maps are especially helpful when dealing with instances in which feasible routing solutions might be difficult to find due to the limitation in the number of available vehicles at each depot. Using these policies, generation of new promising maps is an extremely fast process. In our implementation, the three policies described above are run and the best routing solution obtained with them is used as the initial base and best solutions (lines 16 to 18).

The ILS process, which is started at step 4 (lines 19 to 42), uses a time-based loop. At each iteration, a perturbation operator (lines 20 to 23) is applied on the current base-allocation map, which has an associated routing solution. Intuitively, this base-allocation configuration can be seen as a colored map with different colors for each depot and its corresponding nodes. The proposed perturbation operator generates a new random colored map in which a given percentage p^* of customers has changed colors according to a biased-randomized process (line 23). That is, each depot selects in turn –following any of the aforementioned policies– a node previously removed from the base-allocation map (line 21). The selection is random but, as explained before, it uses the geometric probability distribution to follow the logic of the priority list. In other words, most customers will keep their current color (assigned depot) and just some of them (a given percentage p^*) will change their color in a reasonable way, i.e. according to its priority list. Each time a new allocation map is generated, the CWS routing procedure is applied to solve the resulting CVRPs (line 24). Note that a fast method is needed at this step in order to avoid spending an excessive time on the routing process of a single colored map –which could be a low quality one–, instead of trying different colored maps to look for the most promising ones. As in any ILS framework, the base and best solutions are conveniently updated. If the solution obtained by means of the perturbation process overcomes the best one so far, a more thorough local search (line 33) is carried out before any update is done. This process may be seen as an intensified exploration around potentially good solutions identified during the search. In our case, it is based on the classical *2-opt* operator defined for the CVRP **(Lin, 1965)**. A demon-based acceptance criterion **(Talbi, 2009)** is also introduced to reduce the risk of getting trapped in a local minimum during the search process (lines 36 to 40). Finally, at step 5 (lines 43 to 55), the best solution found so far using a fast routing heuristic is improved by using an efficient routing metaheuristic. Since the metaheuristic might noticeably reduce the routing costs provided by the heuristic, it is recommended to apply the metaheuristic not only to the best-found solution, but also to a reduced list of the top solutions found so far –it might happen, for instance, than the second-ranked solution in the reduced list of candidates improves the first-ranked one after applying the metaheuristic. In our approach, we employ the SR-

GCWS-CS method **(Juan et al., 2009, 2011b)**. The main structure of this method is summarized in **Pseudo-code 6**. The SR-GCWS-CS algorithm also makes use of a biased-randomization strategy plus two specific local search processes. Details on these local search processes can be found in the aforementioned reference. Note, however, that any other efficient routing algorithm could be used here to improve the 'promising' solutions obtained during the ILS stage. Eventually, the ILS-MD procedure will return the best solution found so far.

```
procedure ILS-MD(nodes, depots, vCap, nVeh, maxRouteDistance, pM, pR, p*)
%  Where 0 < pM, pR < 1 represent parameters in two geometric distributions,
%  and 0 < p* < 100 sets the % of nodes to be extracted during the perturbation stage.
%   1. COMPUTE A PRIORITY LIST OF NODES FOR EACH DEPOT.
%   For each depot, compute its sorted list of "priority" nodes.
01  for each depot in depots do
02    priorityList(depot) <- calcPriorityList(nodes, depots, depot)
03  end for
%   2. GENERATE A FEASIBLE INITIAL SOLUTION.
04  maxDemand <- vCap * nVeh  % max demand any depot can satisfy
05  feasible <- false
06  while {feasible is false} do
      % 2.1 Generate a new map in which priority lists are biased-randomized.
07    map <- emptyMap  % reset map for each depot
08    freeNodes <- copy(nodes)
09    randPriorityList <- biasedRand(priorityList, pM)  % use a geometric distribution
10    map <- generateNewMap(map, freeNodes, randPriorityList, maxDemand)
      % 2.2 Try computing a feasible routing solution for the new map.
11    sol <- tryGenFeasibleSol(map, vCap, nVeh, maxRouteDistance)
12    if {sol is not empty} then
13      feasible <- true
14    end if
15  end while
%   3. UPDATE BASE AND BEST SOLUTIONS.
16  baseSol <- sol
17  baseMap <- getMap(baseSol)
18  bestSol <- baseSol
%   4. START AN ITERATIVE LOCAL SEARCH PROCESS.
19  while {termination condition is not met} do  % e.g. time-based condition
      % 4.1 Perturbation stage with a destruction-construction process.
20    map <- extractNodes(baseMap, p*, pM)
21    freeNodes <- substract(getNodes(map), nodes)  % nodes not in map
22    randPriorityList <- biasedRand(priorityList, pM)
23    map <- generateNewMap(map, freeNodes, randPriorityList, maxDemand)
      % 4.2 Try computing a feasible routing solution for the new map.
24    sol <- tryGenFeasibleSol(map, vCap, nVeh, maxRouteDistance)
25    if {sol is not empty} then
26      map <- getMap(sol)
        % 4.3 Update baseSol and bestSol if appropriate.
27      delta <- cost(sol) - cost(baseSol)
28      if {delta < 0} then
29        baseMap <- map
30        baseSol <- sol
31        credit <- delta * (-1)
32        if {cost(sol) < cost(bestSol)} then
33          bestSol <- fastLocalSearch(sol)
34          topSols <- add(bestSol, topSols)  % list of top solutions
35        end if
36      else if {delta > 0} and {delta <= credit} then  % demon-based acceptance.
37        credit <- 0
38        baseMap <- map
39        baseSol <- sol
40      end if
41    end if
42  end while
%   5. APPLY A ROUTING-REFINEMENT PROCESS TO THE TOP SOLUTIONS.
43  for each sol in topSols do
44    for each depot in depots do
45      subSol <- getSubSol(depot, sol)
46      subMap <- getMap(subSol)
47      newSubSol <- SR-GCWS-CS(subMap, vCap, maxRouteDistance, pR)
48      if {cost(newSubSol) < cost(subSol)} and {nVehIn(newSubSol) <= nVeh} then
49        sol <- update(depot, newSubSol, sol)
50      end if
51    end for
52    if {cost(sol) < cost(bestSol)} then
53      bestSol <- sol
54    end if
55  end for
56 return bestSol
end procedure
```

Pseudo-code 2: main ILS-MD procedure.

```
procedure calcPriorityList(nodes, depots, depot)
01  for each node in nodes do
      % Compute distance marginal-savings w.r.t. the best alternative assignment.
02    ms(node) <- dist(node, alternativeDepot(node)) - dist(node, depot)
03    priorityList <- add(ms(node), priorityList)
04  end for
    % Sort priorityList from highest to lowest marginal savings.
05  priorityList <- sort(priorityList)
06  return priorityList
end procedure
```

Pseudo-code 3: procedure to calculate the priority list of each depot.

```
procedure generateNewMap(map, freeNodes, randPriorityList, maxDemand)
01  depots <- getDepots(map)
02  for each depot in depots do
03    subMap(depot) <- getSubMap(map, depot)  % during perturbation, subMap is not empty
04  end for
    % use the least-loaded-depot-first criterion to assign nodes to depots
05  while {freeNodes is not empty} do
06    depot <- selectLeastLoadedDepot(depots) % other criteria are also possible
07    nextNode <- selectNextNode(freeNodes, randPriorityList(depot))
08    subMap(depot) <- add(nextNode, subMap(depot))
09    freeNodes <- delete(nextNode, freeNodes)
10  end while
    % update the subMaps into the map
11  for each depot in depots do
12    map <- update(subMap(depot), map)
13  end for
14  return map
end procedure
```

Pseudo-code 4: procedure to generate a new allocation map.

```
procedure tryGenFeasibleSol(map, vCap, nVeh, maxRouteDistance)
01  sol <- emptySol
02  depots <- getDepots(map)
03  for each depot in depots do
04    subMap <- getSubMap(depot)
05    if {subMap is not in cache} then  % use a hash table cache
        % use the fast CWS heuristic with maxRouteDistance
06      subSol <- cwsAlg(subMap, vCap, maxRouteDistance)
07    else  % use the fast biased-randomized version of the CWS
08      subSol <- randCWS(subMap, vCap, maxRouteDistance, pR)
09    end if
10    if {nVehIn(subSol) > nVeh} then  % no feasible solution found
11       return emptySol
12    else
13       sol <- add(subSol, sol)
14       subSol <- checkCache(subSol, nVeh)  % either improve subSol or update cache
15    end if
16  end for
17  return sol
end procedure
```

Pseudo-code 5: procedure to generate a new solution from a map.

```
procedure SR-GCWS-CS(subMap, vCap, maxRouteDistance, pR)
%  This routing procedure is detailed in Juan et al. (2011b).
01   nodes <- getNodes(subMap)
02   depot <- getDepot(subMap)
     % Compute an initial solution using the CWS heuristic.
03   cwsSol <- cwsAlg(subMap, vCap, maxRouteDistance)
04   bestSol <- cwsSol
05   nIter <- 0
06   while {termination condition is not met} do
       % Generate a newSol using a biased-randomized version of the CWS heuristic.
07     newSol <- randCWS(subMap, vCap, maxRouteDistance, pR)
       % Improve newSol using two different local search processes.
08     newSol <- routeCache(newSol)   % either improve newSol or cache of routes.
09     if {cost(newSol) < cost(cwsSol)} then   % only for "promising" solutions.
10       newSol <- splitting(newSol)   % divide-and-conquer strategy.
11       if {cost(newSol) < cost(bestSol)} then
12         bestSol <- newSol
13       end if
14     end if
15   end while
16   return bestSol
end procedure
```

Pseudo-code 6: summarized version of the SR-GCWS-CS routing algorithm.

6. COMPUTATIONAL EXPERIMENTS

The methodology described in this paper has been implemented as a Java application. Being an interpreted language, Java-based programs do not execute as fast as other compiled programs, such as those developed in C or C++. Nevertheless, Java permits a rapid, platform-independent, development of object-oriented prototypes that can be used to test the potential of an algorithm. A standard personal computer, Intel QuadCore i5 CPU at 3.2 GHz and 4 GB RAM with Windows XP, was used to perform all tests.

In order to test the efficiency of the proposed method and compare it with other existing approaches, we use 33 MDVRP benchmark instances described in **Cordeau et al. (1997)** – instances *p01* to *p23*– and in **Pisinger and Ropke (2007)** –instances *pr01* to *pr10*. These classical instances consider a limited fleet of available vehicles at each depot. It should be noted that some of the instances have a maximum route length allowed and that, for those cases, routes are only allowed to be merged in the savings algorithm as far as this maximum route length is not exceeded. Each instance was run 5 times with a maximum number of iterations of 300,000. After some preliminary tests, the parameter for the pseudo-geometric distribution related to the allocation problem is randomly chosen in the interval *(0.5, 0.8)*. Similarly, the corresponding parameter for the routing algorithm is randomly chosen within the interval *(0.1, 0.2)*. Finally, the degree of perturbation at each iteration of the ILS is selected at random in the range *10% - 50%*. Notice that we only performed a rough and quick fine-tuning process for the aforementioned parameters since we wanted to test if our algorithm was able to provide

competitive results without investing too much time calibrating the parameters. The results of these tests are summarized in **Table 1**.

Table 1: Results obtained for 33 MDVRP benchmark instances.

INSTANCE						OTHER APPROACHES			OUR ILS APPROACH			
Inst.	**n**	**m**	**d**	**Max Route Length**	**Q**	**CGL97 (1)**	**PR07 (2)**	**VCGLR11 (3)**	**OBS (4)**	**Gap (%) (1)-(4)**	**Gap (%) (2)-(4)**	**Gap (%) (3)-(4)**
p01	50	4	4	n/a	80	576.87	576.87	576.87	576.87	0.00%	0.00%	0.00%
p02	50	2	4	n/a	160	473.53	473.53	473.53	473.87	0.07%	0.07%	0.07%
p03	75	3	5	n/a	140	641.19	641.19	641.19	641.19	0.00%	0.00%	0.00%
p04	100	8	2	n/a	100	1001.59	1001.04	1001.04	1003.45	0.19%	0.24%	0.24%
p05	100	5	2	n/a	200	750.03	751.26	750.03	751.9	0.25%	0.09%	0.25%
p06	100	6	3	n/a	100	876.5	876.7	876.5	876.5	0.00%	-0.02%	0.00%
p07	100	4	4	n/a	100	885.8	881.97	881.97	885.19	-0.07%	0.37%	0.37%
p08	249	14	2	310	500	4437.68	4387.38	4372.78	4409.23	-0.64%	0.50%	0.83%
p09	249	12	3	310	500	3900.22	3873.64	3858.66	3882.58	-0.45%	0.23%	0.62%
p10	249	8	4	310	500	3663.02	3650.04	3631.11	3646.67	-0.45%	-0.09%	0.43%
p11	249	6	5	310	500	3554.18	3546.06	3546.06	3547.09	-0.20%	0.03%	0.03%
p12	80	5	2	n/a	60	1318.95	1318.95	1318.95	1318.95	0.00%	0.00%	0.00%
p13	80	5	2	200	60	1318.95	1318.95	1318.95	1318.95	0.00%	0.00%	0.00%
p14	80	5	2	180	60	1360.12	1360.12	1360.12	1360.12	0.00%	0.00%	0.00%
p15	160	5	4	n/a	60	2505.42	2505.42	2505.42	2511.92	0.26%	0.26%	0.26%
p16	160	5	4	200	60	2572.23	2572.23	2572.23	2573.78	0.06%	0.06%	0.06%
p17	160	5	4	180	60	2709.09	2709.09	2709.09	2709.09	0.00%	0.00%	0.00%
p18	240	5	6	n/a	60	3702.85	3702.85	3702.85	3702.85	0.00%	0.00%	0.00%
p19	240	5	6	200	60	3827.06	3827.06	3827.06	3840.91	0.36%	0.36%	0.36%
p20	240	5	6	180	60	4058.07	4058.07	4058.07	4063.64	0.14%	0.14%	0.14%
p21	360	5	9	n/a	60	5474.84	5474.84	5474.84	5574.63	1.82%	1.82%	1.82%
p22	360	5	9	200	60	5702.16	5702.16	5702.16	5737.04	0.61%	0.61%	0.61%
p23	360	5	9	180	60	6095.46	6078.75	6078.75	6084.32	-0.18%	0.09%	0.09%
Average gap p01 – p23										*0.08%*	*0.21%*	*0.27%*
pr01	48	1	4	500	200	-	861.32	861.32	861.32	-	0.00%	0.00%
pr02	96	2	4	480	195	-	1307.34	1307.34	1307.34	-	0.00%	0.00%
pr03	144	3	4	460	190	-	1806.53	1803.80	1803.80	-	-0.15%	0.00%
pr04	192	4	4	440	185	-	2060.93	2058.31	2072.10	-	0.54%	0.67%
pr05	240	5	4	420	180	-	2337.84	2331.20	2342.20	-	0.19%	0.47%
pr06	288	6	4	400	175	-	2685.35	2676.30	2687.73	-	0.09%	0.43%
pr07	72	1	6	500	200	-	1089.56	1089.56	1089.56	-	0.00%	0.00%
pr08	144	2	6	475	190	-	1664.85	1664.85	1668.08	-	0.19%	0.19%
pr09	216	3	6	450	180	-	2136.42	2133.20	2133.56	-	-0.13%	0.02%
pr10	288	4	6	425	170	-	2889.49	2868.26	2889.70	-	0.01%	0.75%
Average gap p01 – p23											*0.07%*	*0.25%*
Average gaps for all problems											*0.17%*	*0.26%*

For each instance, the information included in this table is the following: instance name, number of customers $|V_c|$, vehicles available at each depot $v_{n+k} \in V_d$, number of depots $|V_d|$, maximum route length allowed *L*, and vehicles maximum capacity *Q*. Afterwards, we present the results obtained by **Cordeau et al. (1997)**, referenced as CGL97, **Pisinger and Ropke (2007)**, marked as PR07, and **Vidal et al. (2012)**, shown as VCGLR11. Finally, the best solutions obtained by means of the proposed approach are presented. The gaps between our best solution (OBS) and the different presented references are also included for comparison. As it can be observed, our approach has been able to match 12 out of the 33 previous best-known solutions. Indeed, the average gaps are reasonably low for both sets of instances (*0.27%* for instances *p1-p23* and *0.25%* for instances *pr1-pr10*). In addition, these results are achieved in acceptable computational times (277 seconds, on average using a Java code) regarding the size of the considered instances, the fact that only one standard computer has been used, and also that only 5 runs of the algorithm have been executed. In other words, our approach is able to provide quite competitive solutions in reasonably low computational times. Notice that the average gap between our approach and the PR07 one is just *0.17%*. In fact, our approach is able to improve several individual solutions provided by the PR07 method (see individual negative gaps). Similarly, the average gap between our approach and the VCGLR11 one is just of *0.26%*. At this point it is important to recall that our approach contains very few parameters, and also that it is relatively easy to understand, implement, and use in practical (real-life) situations. On the contrary, the PR07 approach contains 14 parameters, while the VCGLR11 employs 5 components –each of them using one or more parameters– and 3 rules which require from a complex and time-consuming fine-tuning process.

Figure 4 presents a visual comparison between the best-known solution and our best solution for the instance *p21*. As it may be observed, our solution presents a more balanced distribution of customers and routes, an often-desirable characteristic in most realistic cases. For example, in the best-known solution all depots service at least one route with a low number of customers. Although some depots have associated short routes, in our solution the travelled distances are in general more equitably distributed. In addition, our solution is more balanced in terms of the demands assigned to each depot. This characteristic is justified because of the use of the less-loaded criterion to choose which depot selects the next node to be allocated.

Finally, we analyzed how the quality of the results generated by our approach vary when considering different levels of computational time and number of parallel agents –or independent runs of the algorithm. These tests were performed on the *p16*, *p20*, *pr02*, and the *pr08* instances. For each instance, 30 runs of the proposed algorithm were run, each of them initialized with a different random seed. The allowed execution time (in seconds) was adjusted according to the instance size. At each selected time interval, the best solution found was

registered. Then, for each combination of time and number of parallel runs, the gap between the best solution found and the best-known one was computed.

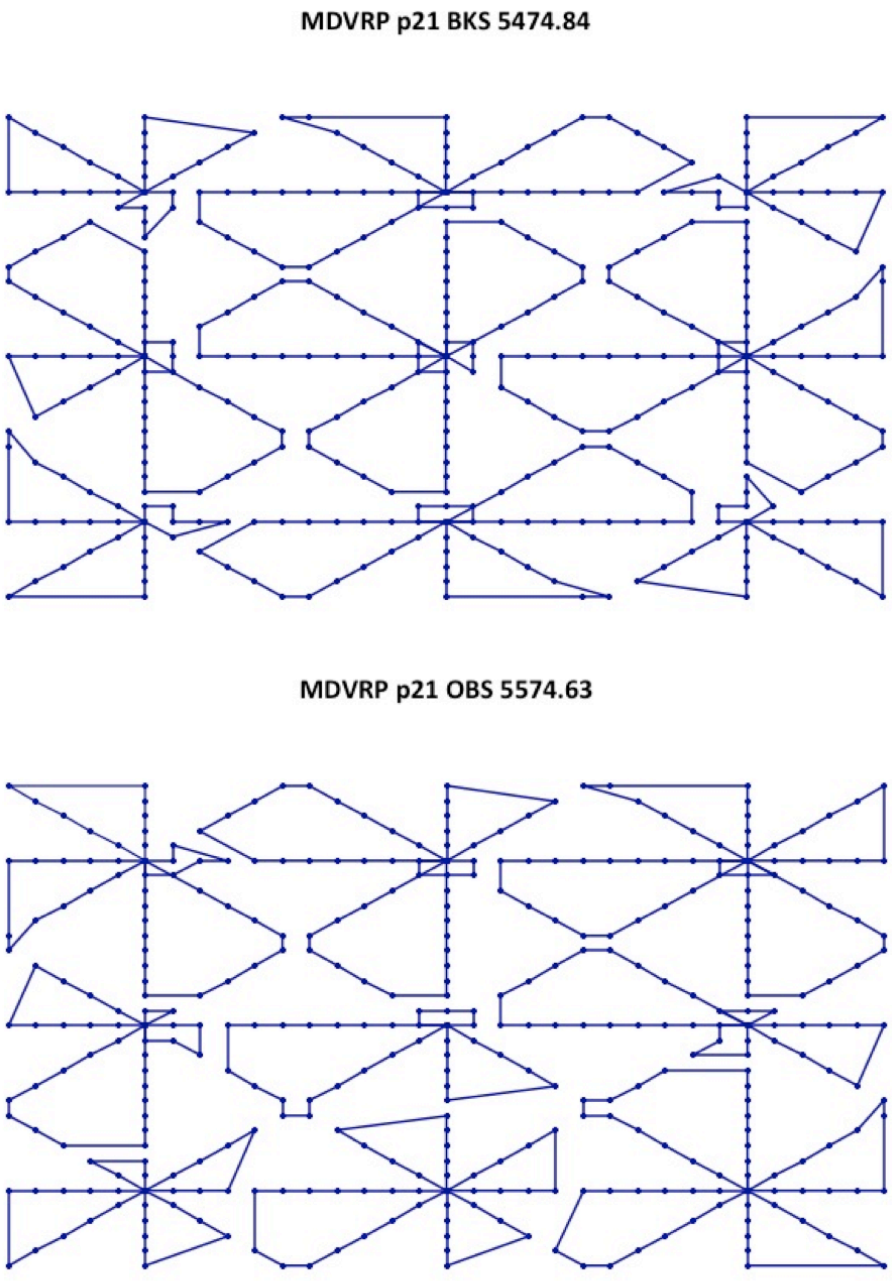
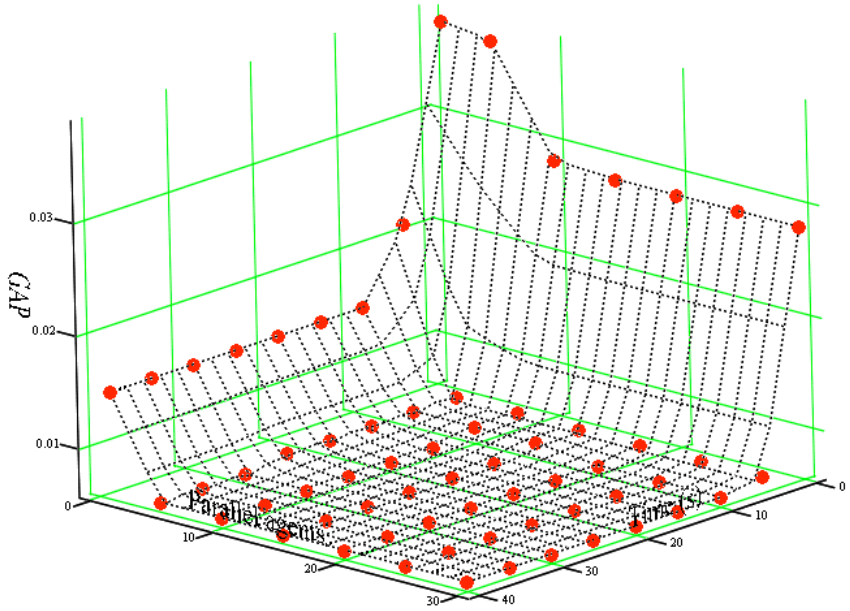


Figure 4: Visual comparison between BKS and OBS for the instance p21.

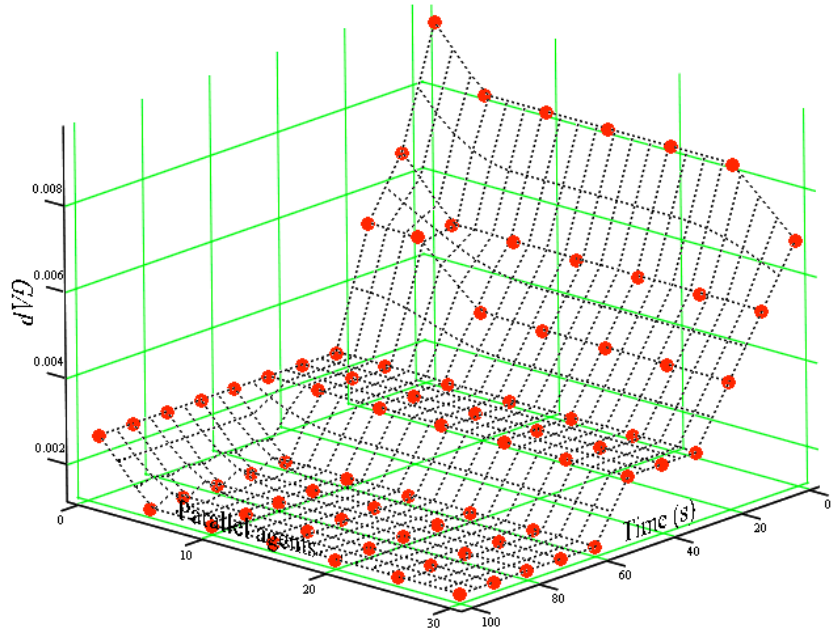
Figures 5 to 8 display the results of this test for the cited instances. Some conclusions may be inferred from the results of these parallelization tests. First, notice that the initial solution we generate as starting point for our algorithm is greatly improved in very short time. Even with one single agent (parallel run with a given random seed) and very few seconds, the gap is significantly decreased. After this steep improvement, the solution evolves more smoothly with

time. In addition, it may be observed that the gap decreases faster as the number of parallel agents grows. Note that using several parallel runs of the algorithm (each of them using a different seed for the random number generator) provides in all cases better results than a single run executed for much more time. In other words, parallelization seems to offer clear benefits to this type of algorithms. In fact, through simple parallelization it is possible to obtain 'high-quality' solutions (average gap below *1%* with respect to the best-known solutions) in almost 'real times' (a few seconds) for all considered instances.



INSTANCE P16

Figure 5: Gap for instance p16, with varying time and number of parallel agents.



INSTANCE P20

Figure 6: Gap for instance p20, with varying time and number of parallel agents.

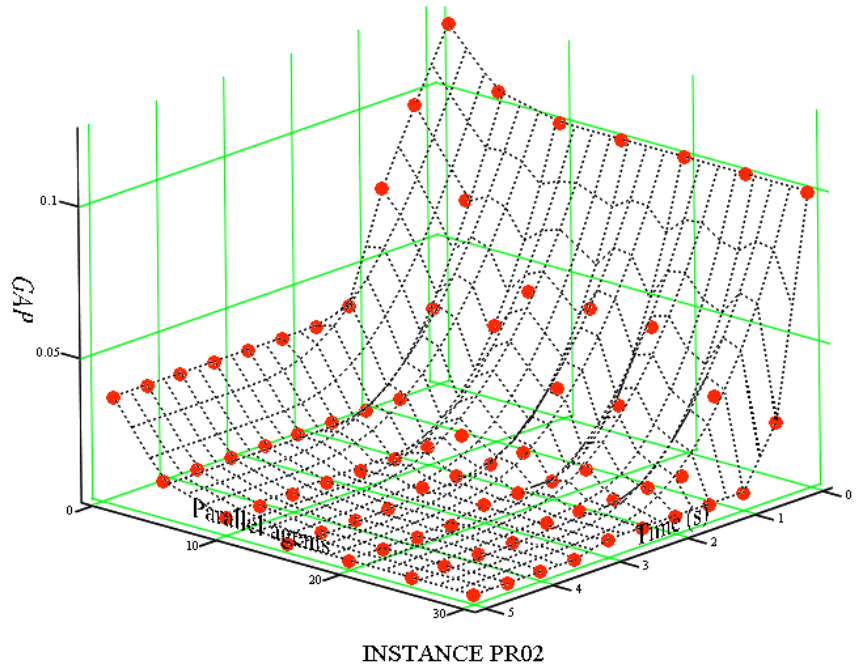


Figure 7: Gap for instance pr02, with varying time and number of parallel agents.

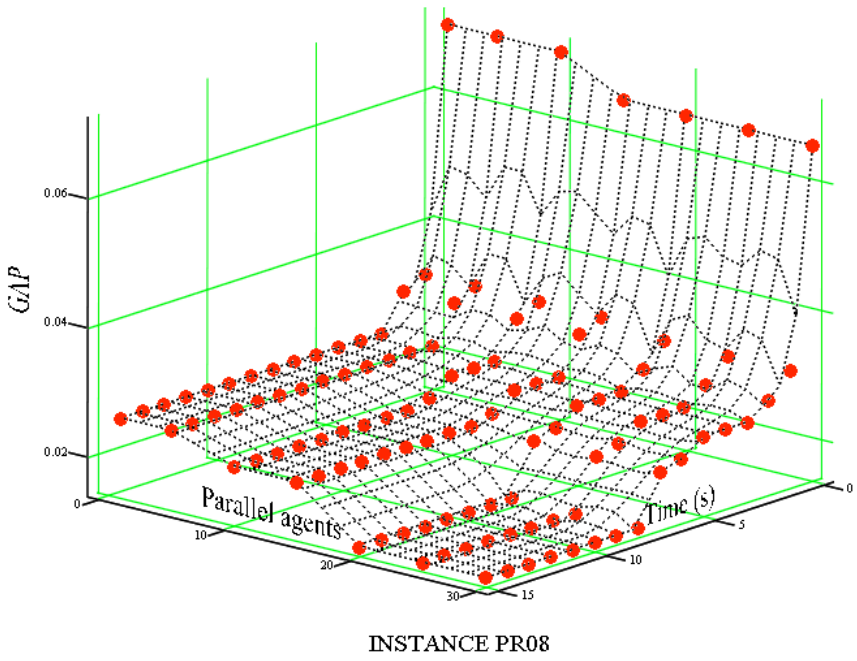


Figure 8: Gap for instance pr08, with varying time and number of parallel agents.

7. CONCLUSIONS

This paper has presented an efficient yet relatively simple approach for solving the Multi-Depot VRP. The proposed method combines biased randomization with an Iterated Local Search metaheuristic framework. As the numerical experiments show, our approach is able to provide

competitive results when compared to other state-of-the-art methods, both in terms of solution quality and computational times.

One of the main advantages of our approach is its relative simplicity: the method is relatively easy to implement and understand, and it needs little fine-tuning process. Only three parameters need to be adjusted: (a) one for the geometric distribution guiding the randomization of the customers-depots allocation process (perturbation operator); (b) one for the geometric distribution guiding the randomization of the routing heuristic (edge-selection process); and (c) the size of the destruction-construction process (also part of the perturbation operator) inside the metaheuristic. In contrast, most state-of-the-art approaches introduce a high number of parameters that should be adjusted, often requiring complex and time-consuming fine-tuning processes. Additionally, the paper contributes to the literature on the use of constructive methods for solving the Multi-Depot VRP. In effect, while most existing metaheuristics for this problem are heavily based on local search, the proposed method relies on biased constructive procedures to generate new solutions. This shows that other similar approaches –e.g. GRASP or Ant Colony Optimization– could also be employed in this field in order to generate competitive results.

The presented methodology is suitable to be run in parallel. By simply changing the values of the seed of the random number generator, several instances of the algorithm can be simultaneously run. As shown in the experiments, this approach leads to 'high-quality' solutions (average gap less than 1% with respect to best-known solutions) in almost 'real-time' (a few seconds). Thus, the aforementioned properties make our approach an excellent alternative for most practical applications of the considered problem. Finally, it is worthy to notice that the combination of Iterated Local Search and biased randomization strategies constitutes an innovative and relatively easy-to-implement framework that can be used to solve a wide range of combinatorial optimization problems.

ACKNOWLEDGMENT

This work has been partially supported by the Spanish Ministry of Science and Innovation (grant TRA2013-48180-C3-3-P), by the CYTED-HAROSA Network (http://dpcs.uoc.edu) and by NICTA. NICTA is funded by the Australian Government through the Department of Communications and the Australian Research Council through the ICT Centre of Excellence Program.

REFERENCES

Bettinelli, A., Ceselli, A., Righini, G., 2011. A branch-and-cut-and-price algorithm for the multi-depot heterogeneous vehicle routing problem with time windows. *Transport Res C-Emer*, 19, 723-740.

Ceselli, A., Righini, G., Salani, M., 2009. A column generation algorithm for a rich vehicle-routing problem. *Transport Sci*, 43(1), 56-69.

Chan, Y., Carter, W.B., Burnes, M.D., 2001. A multiple-depot, multiple-vehicle, location-routing with stochastically processed demands. *Comput Oper Res*, 28, 803-826.

Chao, I.M., Golden, B.L., Wasil, E., 1993. A new heuristic for the multi-depot vehicle routing problem that improves upon best-known solution. *Am J Math Manage Sci*, 13, 371-406.

Clarke, G., Wright, J. 1964. Scheduling of vehicles from a central depot to a number of delivering points. *Operations Research*, 12, 568-581.

Cordeau, J.F., Gendreau, M., Laporte, G., 1997. A tabu search heuristic for periodic and multi-depot vehicle routing problems. *Networks*, 30, 105-119.

Cordeau, J.F., Maischberger, M., 2012. A parallel iterated tabu search heuristic for vehicle routing problems. *Comput Oper Res 2012;* 39, 2033-2050.

Crevier, B., Cordeau, J.F., Laporte, G., 2007. The multi-depot vehicle routing problem with inter-depot routes. *Eur J Oper Res,* 176, 756-773.

Dominguez, O., Juan, A., Faulin, J., 2014. A biased-randomized algorithm for the two-dimensional vehicle routing problem with and without items rotation. *Int. Transactions in Operational Research*, *in press*.

Dondo, R., Cerdá, J., 2007. A cluster-based optimization approach for the multi-depot heterogeneous fleet vehicle routing problem with time windows. *Eur J Oper Res*, 176, 1478-1507.

Dondo, R., Cerdá, J., 2009. A hybrid local improvement algorithm for large-scale multi-depot vehicle routing problems with time windows. *Comput Chem Eng*, 33, 513-530.

Gillett, B.E., Johnson, J.G., 1976. Multi-terminal vehicle-dispatch algorithm. *Omega*, 4, 711-718.

Golden, B., Raghavan, S., Wasil, E., 2008. *The vehicle routing problem: latest advances and new challenges*. Heidelberg: Springer.

Golden, B.L., Magnanti, T.L., Nguyen, H.Q., 1977. Implementing vehicle routing algorithms. *Networks*, 7(2), 113-148.

Guimarans, D., Herrero, R., Riera, D., Juan, A.A., Ramos, J.J., 2011. Combining probabilistic algorithms, constraint programming and lagrangian relaxation to solve the vehicle routing problem. *Ann Math Artif Intel*, 62(3), 299-315.

Ho, W., Ho, G.T.S., Ji, P., Lau, H.C.W., 2008. A hybrid genetic algorithm for the multi-depot vehicle routing problem. *Eng Appl Artif Intel*, 21, 548-557.

Juan, A., Faulin, J., Grasman, S., Riera, D., Marull, J., Mendez, C., 2011a. Using safety stocks and simulation to solve the vehicle routing problem with stochastic demands. *Transport Res C-Emer*, 19, 751-765.

Juan, A., Faulin, J., Jorba, J., Caceres, J., Marques, J., 2013. Using Parallel & Distributed Computing for Solving Real-time Vehicle Routing Problems with Stochastic Demands. Annals of Operations Research, 207: 43-65.

Juan, A., Faulin, J., Jorba, J., Riera, D., Masip, D., Barrios, B, 2011b. On the use of Monte Carlo simulation, cache and splitting techniques to improve the Clarke and Wright Savings heuristic. *Journal of the Operational Research Society*, 62, 1085-1097.

Juan, A., Faulin, J., Ruiz, R., Barrios, B., Gilibert, M., Vilajosana, X, 2009. Using oriented random search to provide a set of alternative solutions to the capacitated vehicle routing problem. In: *Operations Research and Cyber-Infrastructure*, pp. 331-346. Springer.

Juan, A., Lourenço, H., Mateo, M., Luo, R., Castella, Q., 2014. Using Iterated Local Search for solving the Flow-Shop Problem: parametrization, randomization and parallelization issues. *Int. Transactions in Operational Research*, 21(1), 103-126.

Kant, G., Jacks, M., Aantjes, C., 2008. Coca-Cola enterprises optimizes vehicle routes for efficient product delivery. *Interfaces*, 38, 40-50.

Laporte, G., 2007. What you should know about the vehicle routing problem. *Naval Research Logistics*, 54(8), 811-819.

Laporte, G., Nobert, Y., Arpin, D., 1984. Optimal solutions to capacitated multidepot vehicle routing problems. *Congr Numer*, 44, 283-292.

Lin, S., 1965. Computer solutions of the traveling salesman problem. *AT&T Tech J*, 44, 2245-2269.

Lourenço, H.R., Martin, O., Stützle, T., 2003. Iterated local search. In: *Handbook of Metaheuristics*, pp. 321-353. Springer.

Lourenço, H.R., Martin, O., Stützle, T., 2010. Iterated local search: framework and applications. In: *Handbook of Metaheuristics*, pp. 363-397. Springer.

Mirabi, M., Fatemi-Ghomi, S.M.T., Jolai, F., 2010. Efficient stochastic hybrid heuristics for the multi-depot vehicle routing problem. *Robot CIM-INT Manuf*, 26, 564-569.

Pisinger. D., Ropke, S. 2007. A general heuristic for vehicle routing problems. *Comput Oper Res*, 34, 2403-2435.

Prins, C., 2004. A simple and effective evolutionary algorithm for the vehicle routing problem. *Computers & Operations Research*, 31(12), 1985-2002.

Raft, O.M., 1982. A modular algorithm for an extended vehicle scheduling problem. *Eur J Oper Res*, 11, 67-76.

Renaud, J., Laporte, G., Boctor, F.F., 1996. A tabu search heuristic for the multi-depot vehicle routing problem. *Comput Oper Res*, 23(3), 229-235.

Talbi, E., 2009. *Metaheuristics: from design to implementation*. New Jersey: John Willey & Sons.

Tillman, F.A., Cain, T.M., 1972. An upperbound algorithm for the single and multiple terminal delivery problem. *Management Science*, 18, 664-682.

Thangiah, S.R., Salhi, S., 2001. Genetic clustering: an adaptive heuristic for the multi-depot vehicle routing problem. *Appl Artif Intell*, 15, 361-383.

Toth, P., Vigo, D., 2002. *The vehicle routing problem*. SIAM Monographs on Discrete Mathematics and Applications.

Vidal, T., Crainic, T.G., Gendreau, M., Lahrichi, N., Rei, W., 2012. A hybrid genetic algorithm for multi-depot and periodic vehicle routing problems. *Operations Research,* 60(3), 611-624.

Wren, A., Holliday, A. 1972. Computer scheduling of vehicles from one or more depots to a number of delivery points. *OR Oper Res Q,* 23, 333-344.