A SIMULATION-BASED METHODOLOGY TO ASSIST DECISION-MAKERS IN REAL VEHICLE ROUTING PROBLEMS

Angel A. Juan, Daniel Riera, David Masip, Josep Jorba
Dep. of Computer Sciences, Open University of Catalonia, Rambla Poble Nou 156, Barcelona, Spain

Javier Faulin
Dept. of Statistics and OR, Public University of Navarra, Campus Arrosadia, Pamplona, Spain

Keywords: Vehicle routing problem, Hybrid algorithms, Heuristics, Simulation, Decision support systems.

Abstract: The aim of this work is to present a simulation-based algorithm that not only provides a competitive solution for instances of the Capacitated Vehicle Routing Problem (CVRP), but is also able to efficiently generate a full database of alternative good solutions with different characteristics. These characteristics are related to solution's properties such as routes' attractiveness, load balancing, non-tangible costs, fuzzy preferences, etc. This double-goal approach can be specially interesting for the decision-maker, since he/she can make use of this algorithm to construct a database of solutions and then send queries to it in order to obtain those feasible solutions that better fit his/her utility function without incurring in a severe increase in costs. In order to provide high-quality solutions, our algorithm combines a CVRP classical heuristic, the Clarke and Wright Savings method, with Monte Carlo simulation using state-of-the-art random number generators. The resulting algorithm is tested against some well known benchmarks and the results obtained so far are promising enough to encourage future developments and improvements on the algorithm and its applications in real-life scenarios.

1 INTRODUCTION

The Capacitated Vehicle Routing Problem (CVRP) is a NP-hard problem in which a set of customers' demands have to be served by a fleet of homogeneous vehicles departing from a depot, which initially holds all available resources. Of course, there are some tangible costs associated with the distribution of these resources from the depot to the customers. In particular, it is usual to explicitly consider in the model costs due to moving a vehicle from one node –customer or depot– to another. The classical goal here consists on determining the optimal set of routes that minimizes those tangible costs under the following set of constraints: (a) all routes begin and end at the depot; (b) each vehicle has a maximum load capacity, which is considered to be the same for all vehicles; (c) each customer has a well-known demand that must be satisfied; (d) each customer is supplied by a single vehicle, and (e) a vehicle can not stop twice at the same customer.

Even when this problem has been studied for decades, it is still attracting a great amount of attention from top researchers worldwide due to its potential applications, both to real-life scenarios and also to the development of new algorithms, optimization methods and meta-heuristics for solving combinatorial problems (Laporte et al., 2000; Toth & Vigo, 2002; Golden et al., 2008). As a matter of fact, different approaches to the CVRP have been explored during the last decades. These approaches range from the use of pure optimization methods, such as linear programming, for solving small- to medium-size problems with relatively simple constraints, to the use of heuristics and meta-heuristics that provide near-optimal solutions for medium and large-size problems with more complex constraints (Laporte, 2007). Most of the methods cited before focus on minimizing an aprioristic cost function –which usually models tangible costs– subject to a set of well-defined and simple constraints. However, real-life problems can be really complex, with intangible costs, fuzzy

A. Juan A., Riera D., Masip D., Jorba J. and Faulin J. (2009).
A SIMULATION-BASED METHODOLOGY TO ASSIST DECISION-MAKERS IN REAL VEHICLE ROUTING PROBLEMS.
In *Proceedings of the 11th International Conference on Enterprise Information Systems - Artificial Intelligence and Decision Support Systems*, pages 212-217
DOI: 10.5220/0002005602120217

constraints and desirable solution properties that are difficult to be modeled (Poot et al., 2002; Kant et al., 2008). In other words, it is not always straightforward to construct an initial model which takes into account all possible costs (environmental costs, work risks, etc.), constraints and desirable solution properties (time or geographical restrictions, balanced work load among routes, solution attractiveness, etc.). For that reason, there is a need for new methods able to provide a large set of alternative near-optimal solutions with different properties, so that decision-makers can choose among different alternative solutions according to their specific needs and preferences, i.e., according to their utility function, which is usually unknown for the researcher. All in all, as some CVRP specialists have pointed out already, there is a need for more simple and flexible methods to solve the problem, methods that can be used to handle the numerous side constraints that arise in practice (Laporte, 2007).

2 OUR APPROACH

In an effort to give response to the abovementioned demands, this paper aims to present a simple yet powerful hybrid algorithm that combines the parallel version of the classical Clarke & Wright savings (CWS) heuristic (Clarke & Wright, 1964) with Monte Carlo simulation (MCS) and state-of-the-art random number generators to produce a set of alternative solutions for a given CVRP instance. Each solution in this set outperforms the CWS heuristic, but it also has its own characteristics and therefore constitutes an alternative possibility for the decision-maker where several side constraints can be considered. Moreover, the best solution provided by the algorithm is competitive, in terms of aprioristic costs, with the best solution found so far by using existing state-of-the-art algorithms, which tend to be more complex and difficult to implement than the method presented in this paper and, in most cases, require parameter fine-tuning or set-up processes.

Buxey (1979) was probably the first author to combine MCS with the CWS algorithm to develop a procedure for the CVRP. This method was revisited by Faulin & Juan (2008), who introduced an entropy function to guide the random selection of nodes. MCS has also been used by other authors to solve the CVRP (Fernández de Córdoba et al., 2000). In our opinion, recent advances in the development of high-quality pseudo-random number generators (L'Ecuyer, 2002) have opened new perspectives as regards the use of Monte Carlo simulation in combinatorial problems. To test how state-of-the-art random number generators can be used to improve existing heuristics and even push them to new efficiency levels, we decided to combine a MCS methodology with one of the best-known classical heuristics for the CVRP, namely the Clarke & Wright Savings method. In particular, we selected the parallel version of this heuristic, since according to Toth & Vigo (2002), it usually offers better results than the corresponding sequential version.

Therefore, our goal here is to develop a methodology that: (a) provides near-optimal solutions to CVRP instances with respect the objective function, and (b) provides the decision-maker with a large set of alternative good solutions for a given CVRP instance, each of them with different characteristics. Once generated, this list of alternative good solutions can be classified and stored in a solutions database so that the decision-maker can perform retrieval queries according to different criteria or preferences regarding the desirable properties of an ideal real-life solution.

In order to develop such a methodology, we introduce some specific random behavior within the CWS heuristic and then start an iterative process with it. This random behavior helps us to start an efficient search process inside the space of feasible solutions. Each of these feasible solutions will consist of a set of roundtrip routes from the depot that, altogether, satisfy all demands of the nodes by visiting and serving all them exactly once. At each step of the solution-construction process, the CWS algorithm always chooses the edge with the highest savings value. Our approach, instead, assigns a probability of selecting each edge in the savings list. According to our design, this probability should be coherent with the savings value associated with each edge, i.e., edges with higher savings will be more likely to be selected from the list than those with lower savings. Finally, this selection process should be done without introducing too many parameters in the methodology –otherwise, it would be necessary to perform fine-tuning processes, which tend to be non-trivial and time-consuming. To reach all those goals, we employ the geometric statistical distribution with parameter α ($0 < \alpha < 1$) during the CWS solution-construction process: each time a new edge hast to be selected from the list of available edges, a geometric distribution is randomly selected. This distribution is then used to assign exponentially diminishing probabilities to each eligible edge according to its position inside the savings list, which has been previously sorted by its

corresponding savings value. That way, edges with higher savings values are always more likely to be selected from the list, but the probabilities assigned are variable and they depend upon the concrete distribution selected at each step. By iterating this procedure, an oriented random search process is started. Notice that this general approach has similarities with the Greedy Randomized Adaptive Search Procedure (GRASP) (Feo & Resende 1995). GRASP is a typically two-phase approach where in the first phase a constructive heuristic is randomized. The second phase includes a local search process. Nevertheless, it is important to notice that our algorithm does not require any adaptive effort –the savings list is calculated just once, at the beginning of the process. Moreover, our approach is strongly based on the combination of a classical heuristic with statistical distributions and Monte Carlo simulation, which is not the usual case in GRASP algorithms.

Next, we describe with more detail the main steps of our approach (Fig. 1):

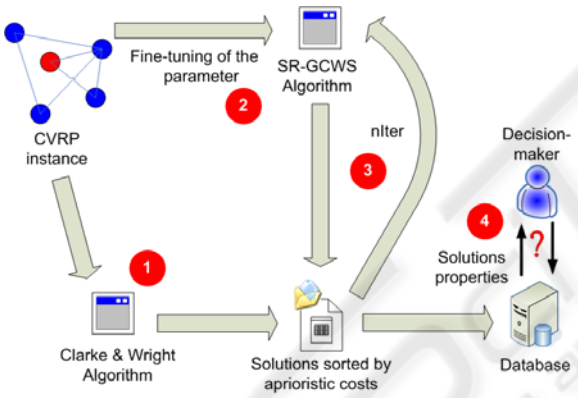

Figure 1: Scheme of our approach for the CVRP.

1. Given a CVRP instance, construct the corresponding data model and use the classical CWS algorithm to solve it.
2. Choose a value for the parameter α for adding random behavior to the algorithm; according to our experience, any parameter value between 0.10 and 0.15 will give promising results in most tested instances, so that no fine-tuning process is really needed.
3. Start an iterative process to generate solutions using the SR-GCWS algorithm with the user-defined values for parameters α and the number of iterations to run (nIter).
4. For each one of the iterations, save the resulting solution in a database only if it outperforms the one provided by the CWS algorithm, i.e., we will consider that a solution is a good one only if it outperforms the CWS solution from an aprioristic costs perspective.

Roughly speaking, if the size of the savings list in the current step is large enough, the parameter α can be interpreted as the probability of selecting the edge with the highest savings value at the current step of the solution-construction process. Choosing a relatively low α-value (e.g. α = 0.05) implies considering a large number of edges from the savings list as potentially eligible, e.g.: assuming a list size of 100, if we choose α = 0.05 then the list of potentially eligible edges will cover about 44 edges from the sorted savings list. On the contrary, choosing a relatively high α-value (e.g. α = 0.35) implies reducing the list of potential eligible edges to just a few of them, e.g.: assuming a list size of 100, if we choose α = 0.35 then the list of potentially eligible edges will be basically reduced to approximately 5 edges.

Once a value for α is chosen, the first edge must be selected. This same value of α can be used for all future steps. In that case, the selection of the α-value might require some minor fine-tuning process. However, based on the tests we have performed so far, we prefer to consider this α-value as a random variable whose behavior is determined by a well-known continuous distribution, e.g.: a uniform distribution in the interval (0.05, 0.20). This way, we do not only avoid a fine-tuning process, but we also have the possibility to combine different values of this parameter at different edge-selection steps of the same solution-construction process. In other words, we are interested in the possibility of combining different strategies regarding the number of edges to be considered as eligible at different steps through the solution-construction process.

3 ALGORITHM DESCRIPTION

In this section, we will discuss the main parts of the SR-GCWS algorithm. Fig. 2 shows the main procedure, which drives the solving methodology. First, inputs are entered in the program and both nodes and constraints are identified. Then, an efficiency list is constructed. This list contains the potential edges to be selected sorted by their associated savings. At this point, the CWS solution is obtained by applying the Clarke & Wright Savings heuristic. The costs associated with this solution will be used as an upper bound limit for the costs of what we will consider a good solution.

Therefore, from this moment on we will save in a database all new solutions that improve those costs. It is at this step when we start the iterative process to generate hundred/thousands of solutions by using our algorithm. Details of this constructive process can be found at pseudo-codes included in Figs. 3 and 4. The former figure shows how the construction process starts with the CWS initial solution and then adds a randomized edge selection before proceeding with the merging of existing routes if possible. The later figure shows some details of the random selection process.

```
procedure SR-GCWS()
1 vrp = getInstanceInputs();
2 nodes = getNodes(vrp);
3 constraints = getConstraints(vrp);
4 effList = buildEfficiencyList(nodes);
5    cwsSol    =    buildCWSSolution(vrp,
effList);
6 while stopping criterion not satisfied→
7  sol  =  buildRandomizedSolution(nodes,
constraints, effList);
8 sol = improveSolUsingHashTable(sol); //
  learn from experience
9 updateSolution(sol,
  bestSolutionFound[]);
10 elihw
11 return bestSolutionFound[];
end
```

Figure 2: Generic SR-GCWS procedure.

```
procedure  buildRandomizedSolution(nodes,
constraints, list)
1 sol = buildTrivialCWSSol(nodes);
2 list = sort(list); // sort edge list
3 while list contains edges →
4    e = selectEdgeAtRandom(list);
5    deleteFromList(e);
6       if CWS route-merging meets all
constraints →
7   sol = insertEdgeInSol(e); //merging
8    fi
9 elihw
10 return sol;
end
```

Figure 3: Constructive process to generate new solutions.

Finally, at each iteration, we try to improve the solution being constructed by saving in a hash table the best routes found so far for any specific set of nodes. The hash table always keeps the best found-so-far way of sorting a set of customers in a route. In some sense, this could be considered as a learning mechanism that makes the algorithm more efficient as more new solutions are generated. A hash table is used instead of an array just for computational efficiency reasons.

```
procedure selectEdgeAtRandom(list)
1 beta = generateRandomNumber(a, b); //
e.g.: a = 0.05 and b = 0.20
2 randomValue = generateRandomNumber(0,
1);
3 n = 0;
4 cumulativeProbability = 0.0;
5 for each edge in list →
6  edgeProbability = beta * (1 – beta)^n;
// geometr. distribution
7          cumulativeProbability      +=
edgeProbability;
8  if randomValue < cumulativeProbability
→
9     return edge;
10 else
11    n++;
12 fi
13 rof
14  k  =  generateIntegerRandomNumber(0,
list size); // from 0 to list size – 1
15 return getEdgeAtPosition(k);
end
```

Figure 4: Randomized selection of edges from savings list.

4 EXPERIMENTAL RESULTS

The methodology described in this paper has been implemented as a Java application. At the core of this application, some state-of-the-art pseudo-random number generators are employed. In particular, we have used some classes from the SSJ library (L'Ecuyer, 2002), among them, the subclass GenF2W32, which implements a generator with a period value equal to 2^{800}-1. Using a pseudo-random number generator with such an extremely long period is especially useful when performing an in-depth random search of the solutions space. In our opinion, the use of such a long-period RNG has other important advantages: the algorithm can be easily parallelized by splitting the RNG sequence in different streams and using each stream in different threads or CPUs. This can be an interesting field to explore in future works, given the current trend in multi-core processors and parallel computing.

In order to verify the goodness of our approach and its efficiency as compared with other existing methodologies, a total of 14 classical CVRP benchmark instances were selected from the reference web site http://www.branchandcut.org, which contains detailed information regarding a large number of benchmark instances. The selection process was based on the following criteria: (a) all 6

Table 1: Comparison of methodologies for six randomly selected CVRP instances.

Instance	Number of nodes	CWS solution	Best-known solution*	Our best solution	Gap
A-n45-k7	45	1,199.98	1,147.28	1,146.91	-0.03%
A-n60-k9	60	1,421.88	1,355.80	1,355.80	0.00%
A-n80-k10	80	1,860.94	1,766.50	1,766.50	0.00%
B-n50-k7	50	748.80	744.78	744.23	-0.07%
B-n52-k7	52	764.90	750.08	749.97	-0.01%
B-n57-k9	57	1,653.42	1,603.63	1,602.29	-0.08%
B-n78-k10	78	1,264.56	1,229.27	1,228.16	-0.09%
E-n51-k5	51	584.64	524.94	524.61	-0.06%
E-n76-k10	76	900.26	837.36	839.13	0.21%
E-n76-k14	76	1,073.43	1,026.71	1,026.14	-0.06%
F-n135-k7	135	1,219.32	1,170.65	1,170.33	-0.03%
M-n121-k7	121	1,068.14	1,045.16	1,045.60	0.04%
P-n70-k10	70	896.86	830.02	831.81	0.22%
P-n101-k4	101	765.38	692.28	691.29	-0.14%
				Average gap	-0.01%

(*) Best-known solution according to the information available at **http://www.branchandcut.org/**

sets of instances (A, B, E, F, M and P) were used, (b) only instances offering complete information (e.g. specific routes in best known solution) were considered, (c) instances with less than 45 nodes were avoided, since they can be easily optimized by using exact methods. The selected benchmark files are shown on Table 1. These instances differ in the number of nodes (ranging from 45 to 135) and also in the location of the depot with respect to the customers (in some cases the depot occupies a central position in the scatterplot of customers while in others the depot is located at one corner). Some instances characterized by having clusters of customers have also been considered.

A standard personal computer, Intel® Core™2 Duo CPU at 2.4 GHz and 2 GB RAM, was used to perform all tests. Results of these tests are summarized in Table 1, which contains the following information for each instance: (a) number of nodes, (b) costs associated with the solution given by the parallel version of the CWS heuristic, (c) costs associated to the best-known-so-far solution, C, according to the information available at the referred web site, (d) costs associated with the best-known-so-far solution provided by our algorithm, C', and finally (e) gap between both solutions calculated as the quotient $(C'-C)/C$ and expressed as a percentage value. Notice that, as defined, a positive gap will imply that our solution costs are higher than the ones associated with the best-known-so-far solution, while a negative gap will imply just the opposite, i.e., better solutions found by our approach.

From Table 1 it can be deduced that, for each of the 14 randomly selected instances, our methodology has been able to provide a virtually equivalent solution to the one considered as the best-known-so-far. In fact, in 9 out of the 14 tested instances, our methodology has been able to slightly improve the best-known solution, offering negative gaps. When considering all instances together, the average gap is still negative (its value is equal to -0.01%). Generally speaking, according to these results, it seems licit to say that the hybrid methodology presented here is able to generate excellent CVRP solutions.

As described before, our approach makes use of an iterative process to generate a set of random feasible solutions. According to the experimental tests carried out, each iteration is completed in just a few milliseconds by using a standard computer. By construction, odds are than the generated solution outperforms the one given by the CWS heuristic. This means that our approach provides, almost immediately (even for the instance with 200 nodes), what we call “a class C solution”, i.e., a feasible solution which outperforms the CWS heuristic in aprioristic costs. Moreover, as verified by testing, hundreds or even thousands of alternative class C solutions can be obtained after some minutes of computation, each of them having different attributes regarding non-aprioristic costs, workload balance, visual attractiveness, etc. By doing so, a list of alternative solutions can be constructed, thus allowing the decision-maker to filter this solutions list according to different criteria. This offers the decision-maker the possibility to choose, among

different solutions with similar aprioristic costs, the one which best fulfils his or her preferences according to his or her utility function.

Furthermore, and again according to experimental results, our algorithm is able to provide a "class B" solution, i.e., a feasible solution inside the 2% gap from the best-known solution, in just some hundred or some thousand iterations, which in small- and medium-size instances takes only a few seconds to run. Of course, as it has been already discussed in the previous section, with more computing time our algorithm is capable to provide "class A" solutions, i.e., feasible solutions that are virtually equivalent, or even better in some cases, to the best-known-so-far solution for every tested instance. Another important point to consider here is the simplicity of the presented methodology. In effect, our algorithm needs little instantiation and does not require any fine-tuning or set-up processes. This is quite interesting in our opinion, since according to (Kant et al., 2008) some of the most efficient heuristics and meta-heuristics are not used in practice because of the difficulties they present when dealing with real-life problems and restrictions. On the contrary, simple hybrid approaches like the one introduced here tend to be more flexible and, therefore, they seem more appropriate to deal with real restrictions and dynamic work conditions. Notice also that our approach seems to work well in all tested instances without requiring any special fine-tuning or set-up process.

Finally, it is convenient to highlight that the introduced methodology can be used beyond the CVRP scenario: with little effort, similar hybrid algorithms based on the combination of Monte Carlo simulation with already existing heuristics can be developed for other routing problems and, in general, for other combinatorial optimization problems. In our opinion, this opens a new range of potential applications that could be explored in future works.

5 CONCLUSIONS

In this paper a simple yet efficient hybrid methodology for solving the Capacitated Vehicle Routing Problem has been presented. This methodology, which does not require any particular fine-tuning or configuration process, combines the classical Clarke & Wright heuristic with Monte Carlo simulation using a geometric distribution and a state-of-the-art pseudo-random number generator. Results show that our methodology is able to provide top-quality solutions which can compete with the ones provided by much more complex meta-heuristics, which usually are difficult to implement in practice. Moreover, being a constructive approach, it can generate hundreds of alternative good solutions in a reasonable time-period, thus offering the decision-maker the possibility to apply different non-aprioristic criteria when selecting the solution that best fits his or her utility function.

REFERENCES

Buxey, G.M. 1979. The Vehicle Scheduling Problem and Monte Carlo Simulation. Journal of Operational Research Society, 30, 563-573

Clarke, G. and J. Wright. 1964. Scheduling of Vehicles from a central Depot to a Number of Delivering Points. Operations Research, 12, 568-581

Faulin, J. and A. Juan. 2008. The ALGACEA-1 Method for the Capacitated Vehicle Routing Problem. International Transactions in Operational Research, 15, 1-23

Feo, T. A, Resende, M.G. C. 1995. Greedy randomized adaptive search procedures. Journal of Global Optimization, 6, 109-133

Fernández de Córdoba, P., García Raffi, L.M., Mayado, A. and J.M. Sanchis. 2000. A Real De-livery Problem Dealt with Monte Carlo Techniques. TOP, 8, 57-71

Golden, B., Raghavan, S. and E. Edward Wasil (eds.). 2008. The Vehicle Routing Problem: Lat-est Advances and New Challenges. Springer

Kant, G., Jacks, M. and C. Aantjes. 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: 811-819

Laporte, G., Gendreau, M., Potvin, J.Y. and F. Semet. 2000. Classical and Modern Heuristics for the Vehicle Routing Problem. International Transactions in Operational Research, 7, 285-300

Law, A. 2007. Simulation Modeling & Analysis. McGraw-Hill

L'Ecuyer, P. 2002. SSJ: A Framework for Stochastic Simulation in Java. In Proceedings of the 2002 Winter Simulation Conference, pp. 234 – 242

Poot, A., Kant, G. and A Wagelmans. 2002. A savings based method for real-life vehicle routing problems. Journal of the Operational Research Society, 53, 57-68

Toth, P. and D. Vigo. 2002. The Vehicle Routing Problem. SIAM Monographs on Discrete Mathematics and Applications. SIAM