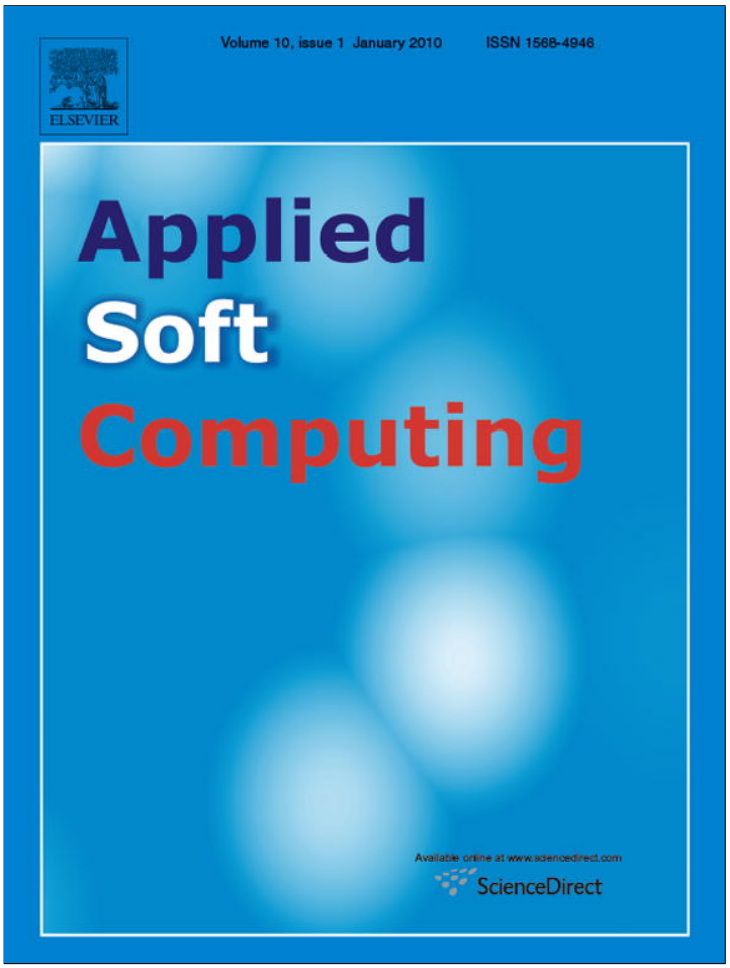

Applied Soft Computing 10 (2010) 215–224

Contents lists available at ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc

The SR-GCWS hybrid algorithm for solving the capacitated vehicle routing problem

Angel A. Juan [a,*], Javier Faulin [b], Rubén Ruiz [c], Barry Barrios [b], Santi Caballé [a]

[a] Dep. of Computer Science and Telecommunication, Open University of Catalonia, Rambla Poblenou, 156, 08018 Barcelona, Spain
[b] Dep. of Statistics and Operations Research, Public University of Navarre, Campus Arrosadia, 31006 Pamplona, Spain
[c] Grupo de Sistemas de Optimización Aplicada - ITI, Universidad Politécnica de Valencia, Camino de Vera, s/n, 46021 Valencia, Spain

ARTICLE INFO

Article history:
Received 13 November 2008
Received in revised form 16 May 2009
Accepted 5 July 2009
Available online 31 July 2009

Keywords:
Hybrid algorithms
Heuristics
Monte Carlo simulation
Capacitated vehicle routing problem

ABSTRACT

The capacitated vehicle routing problem (CVRP) is a well known problem which has long been tackled by researchers for several decades now, not only because of its potential applications but also due to the fact that CVRP can be used to test the efficiency of new algorithms and optimization methods. The objective of our work is to present SR-GCWS, a hybrid algorithm that combines a CVRP classical heuristic with Monte Carlo simulation using state-of-the-art random number generators. The resulting algorithm is tested against some well-known benchmarks. In most cases, our approach is able to compete or even outperform much more complex algorithms, which is especially interesting if we consider that our algorithm does not require any previous parameter fine-tuning or set-up process. Moreover, our algorithm has been able to produce high-quality solutions almost in real-time for most tested instances. Another important feature of the algorithm worth mentioning is that it uses a randomized constructive heuristic, capable of generating hundreds or even thousands of alternative solutions with different properties. These alternative solutions, in turn, can be really useful for decision-makers in order to satisfy their utility functions, which are usually unknown by the modeler. The presented methodology may be a fine framework for the development of similar algorithms for other complex combinatorial problems in the routing arena as well as in some other research fields.

1. Introduction

In the capacitated vehicle routing problem (CVRP) a set of customer demands have to be served with a fleet of vehicles from a depot or central node. Each vehicle has the same capacity (homogeneous fleet) and each customer has a certain demand that must be satisfied. Additionally, there is a cost matrix that measures the costs associated with moving a vehicle from one node to another. These costs usually represent distances, traveling times, number of vehicles employed or a combination of these factors.

More formally, we assume a set Ω of $n+1$ nodes, one of which represents the vehicle origin and destination (depot node) and the rest of which the delivery points (demanding nodes). The nodes are numbered from 0 to n, node 0 being the depot and the remaining n nodes the delivery points. A demand $q_i > 0$ of some commodity has been assigned to each non-depot node i ($1 \leq i \leq n$). On the other hand, $E = \{(i,j)/i,j \in \Omega; i < j\}$ represents the set of the $n(n+1)/2$ existing edges connecting the $n+1$ nodes. Each of these links has an associated aprioristic cost, $c_{ij} > 0$, which represents the cost of sending a vehicle from node i to node j. These c_{ij} are assumed to be symmetric ($c_{ij} = c_{ji}$, $0 \leq i,j \leq n$), and they are frequently expressed in terms of the Euclidean distance, d_{ij}, between the two nodes. The delivery process is to be carried out by a fleet of NV vehicles ($NV \geq 1$) with equal capacity, $C \gg \max\{q_i/1 \leq i \leq n\}$. Some additional constraints associated with the CVRP are the following [32]:

1. Each non-depot node is supplied by a single vehicle,
2. All vehicles begin and end their routes at the depot (node 0),
3. A vehicle cannot stop twice at the same non-depot node,
4. No vehicle can be loaded exceeding its maximum capacity.

Different approaches to the CVRP have been explored during the last decades [53,22]. These approaches range from the use of pure optimization methods, such as linear programming, for solving small-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

* Corresponding author.
E-mail addresses: ajuanp@uoc.edu (A.A. Juan), javier.faulin@unavarra.es (J. Faulin), rruiz@eio.upv.es (R. Ruiz), bcubeb3@mit.edu (B. Barrios), scaballe@uoc.edu (S. Caballé).

1568-4946/$ – see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.asoc.2009.07.003

constraints. Most of these methods focus on minimizing an aprioristic cost function subject to a set of well-defined constraints. However, real-life problems tend to be complex enough so that not all possible costs, e.g., environmental costs, work risks, etc., constraints and desirable solution properties, e.g., time or geographical restrictions, balanced work load among routes, solution attractiveness, etc., can be considered a priori during the mathematical modeling phase [46,27]. For that reason, there is a need for more flexible 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 necessities and preferences, i.e., according to their utility function, which is usually unknown for the researcher. Furthermore, in a recent critical review by Laporte [29], the author clearly states "When reporting results, most researchers concentrate on solution quality and computing time. While these two measures are undoubtedly important, they do not tell the whole story. Other qualities such as simplicity of implementation and flexibility are also important (...) It is also important to design algorithms that can easily handle the numerous side constraints that arise in practice". Consequently, the main purpose of this paper is to present SR-GCWS (Simulation in Routing via the Generalized Clarke and Wright Savings heuristic), a hybrid algorithm that combines the parallel version of the classical Clarke and Wright Savings (CWS) heuristic 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 some cases, require parameter fine-tuning or set-up processes.

The rest of the paper is structured as follows: Section 2 is dedicated to background and literature review; Section 3 presents a global intuitive description of our proposed approach; Section 4 explains some technical details; Section 5 shows the pseudo-code associated with the proposed algorithm; Section 6 analyzes the suitability of our approach by carrying out several experimental tests regarding some very well known benchmark files; Section 7 discusses the main contributions of our work. Finally, Section 8 summarizes and concludes the paper.

2. Background and literature review

As with many other Operations Research problems, CVRP research started with exact methodologies like linear programming, dynamic programming or branch and bound algorithms. Some noteworthy examples are the branch and cut methods in [41,17,2]. In [2] the authors are able to reach decent levels of performance for exact methods, where exact solutions to problems up to 100 customers are obtained. However, the success rate is variable [29]. Furthermore, the computational times are extreme in some cases and adding new real constraints is a challenge in such specialized exact methodologies. Recently, some authors have been able to extend some formulations and lower bounds to a set of different variants of the VRP [3]. As far as we know, this is a first attempt at a more general exact framework for routing problems. Reviews of exact techniques for VRP problems are available [30,52]. Unfortunately, exact techniques to more complex VRP variants are not so well performing. For example, the well-known VRP with time windows or VRPTW is significantly more challenging as far as exact methods are concerned. Accordingly, the state-of-the-art results presented in [10,26], among many others, show techniques that solve problems with far less customers.

The literature is also very rich on heuristic approaches for the VRP, largely motivated by the limited results obtained by exact techniques in large realistic problems. Clarke and Wright's Savings (CWS) constructive algorithm [8] is probably the most cited heuristic to solve the CVRP. This procedure uses the concept of savings. Generally speaking, at each step of the solution-construction process, the edge with the most savings is selected if and only if the two corresponding routes can feasibly be merged and if the selected edge comprises of nodes that are not interior to its respective route (a node is interior to a route if it is not adjacent to the depot). The CWS algorithm usually provides relatively good solutions, especially for small and medium-size problems, but it also presents difficulties in some cases [18]. Many variants and improvements of the CWS have been proposed in the literature. For instance, [40] generalized the definition of the savings function, introducing two parameters for controlling the savings behavior. Similarly, other authors developed a procedure based upon the CWS algorithm, using the same savings function but introducing a solution perturbation scheme in order to avoid poor quality routes [23]. In [5], the CWS method was adapted in order to use it to optimize inter-customer travel times. Correspondingly, in [11] a version of the CWS method for the Stochastic VRP can be found. Two years later, in [44] the main characteristics of the CWS method and its performance in generic VRP were analyzed. Recently, the CWS heuristic has been finely tuned by means of genetic algorithms experimentation [4]. For a more comprehensive discussion on the various CWS variants, the reader is referred to [31,29].

Another important approach to the CVRP is the so-called petal methods, starting with the most basic sweep method [21] and its extensions proposed by many authors. As noted in [29], these petal-based algorithms like those in [49,48] are, on average, better performers than CWS-based methodologies developed so far.

Using constructive heuristics as a basis, meta-heuristics became popular for the VRP during the nineties. Some early examples are the Tabu Route method [19] or the Boneroute method [51]. Tabu search algorithms, like those proposed in [50,54] are among the most popular meta-heuristics. Genetic algorithms have also played a major role in the development of effective approaches for the VRP. Some examples can be found in [1,6,47]. More recent works are those presented in [37,38], among many others. Advanced crossover operators are put forward in [42].

Large sized problems can be efficiently solved by means of variable neighborhood search (VNS) methods as shown in [28]. Similarly, other authors proposed the use of general local search methods working over adaptive large neighborhoods [45].

As we can see, the literature on the VRP is vast and large. We have also cited many recent papers, which demonstrate that the VRP is still a very active and proficient research field. Obviously, due to space limitations, a complete review of the vast VRP literature is not given here. For more detailed reviews, the reader is referred to [9,29,20].

The methodology we present in this paper combines the CWS algorithm with the use of Monte Carlo simulation (MCS), which can be defined as a set of techniques that make use of random numbers and statistical distributions to solve certain stochastic or deterministic problems [33]. MCS has proved to be extremely useful for obtaining numerical solutions to complex problems which cannot be efficiently solved by using analytical approaches. Buxey [7] was probably the first author to combine MCS with the CWS algorithm to develop a procedure for the CVRP. This method was revisited in

[12], who introduced an entropy function to guide the random selection of nodes. MCS has also been used in [16,13,24,25] to solve the CVRP.

3. Designing the SR-GCWS algorithm

In our opinion, recent advances in the development of high-quality pseudo-random number generators [35] have opened new perspectives as regards the use of Monte Carlo simulation (MCS) 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 and Wright Savings (CWS) method. In particular, we selected the parallel version of this heuristic as described in [39], since it usually offers better results than the corresponding sequential version [53].

So, our aim was to introduce some nice random behavior within the CWS heuristic in order 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. As stated in Section 2, 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. Moreover, 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 different geometric statistical distributions during the CWS solution-construction process: each time a new edge must be selected from the list of available edges, a (quasi-) geometric distribution is randomly selected (details are given in the next section); this distribution is then used to assign (quasi-) 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 exact probabilities assigned are variable and they depend upon the concrete distribution selected at each step. By iterating this methodology, a random but efficient search process is started. Notice that this general approach has similarities with the Greedy Randomized Adaptive Search Procedure (GRASP) [14,15]. GRASP is a typically two-phase approach where in the first phase a constructive heuristic is randomized. The second phase includes a local search phase. Our proposed approach does without the expensive local search phase and includes a more detailed randomized construction step. By doing so we have a more general and less instantiated method as local search needs to be instantiated for every different problem.

4. A more formal description of the edge selection process

As we have explained in the previous section, during the solution-construction process, each time a new edge needs to be selected from the savings list, a different quasi-geometric distribution is chosen. For each edge in the savings list, this distribution defines its probability of being selected at the current edge-selection step of the process. More precisely, each time a new edge must be selected, we choose a real value α, $0 < \alpha < 1$, and then consider the following probability distribution for the random variable X = "node k-th is selected at the current step", where $k = 1,2,\ldots,s$, with s being the current size of the list:

$$P(X = k) = \alpha \cdot (1-\alpha)^{k-1} + \varepsilon \quad \forall\, k = 1, 2, \ldots, s$$

where

$$\varepsilon = \sum_{k=s+1}^{+\infty} \alpha \cdot (1-\alpha)^{k-1} = 1 - \sum_{k=1}^{s} \alpha \cdot (1-\alpha)^{k-1}$$

Even when the distribution defined here is clearly inspired on the geometric one, the latter assigns a positive probability to every value in the interval $[1, +\infty)$ and, therefore does not consider the error term ε. Because of this, we classify the former distribution as a quasi-geometric distribution.

Notice that the list size s diminishes as the process evolves and new edges are extracted from it. Roughly speaking, if the size of the savings list in the current step is large enough, the term ε is close to zero and, therefore, 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. As Fig. 1 shows, choosing a relatively low α-value (e.g., $\alpha = 0.05$) implies considering a large number of edges from the savings list as potentially eligible, e.g.: assuming $s = 100$, if we choose $\alpha = 0.05$ then the list of potentially eligible edges will cover about 44 edges

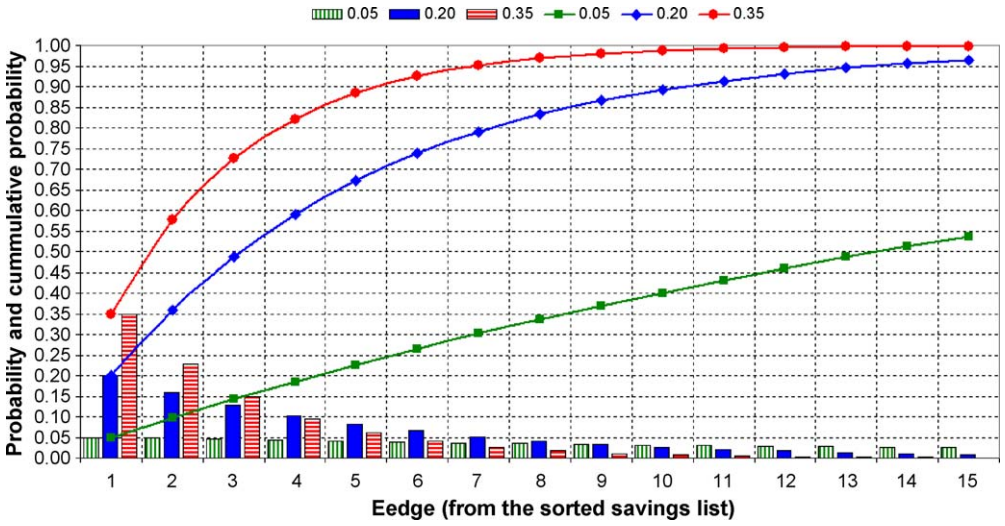

Fig. 1. Effects of the chosen parameter on the geometric distribution.

```
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); // cache of routes
9   updateSolution(sol, bestSolutionFound[]);
10 end while
11 return bestSolutionFound[];
end
```

Fig. 2. Main procedure.

from the sorted savings list, since $P(X \leq 44) \approx 0.9$. On the contrary, choosing a relatively high α-value (e.g., $\alpha = 0.35$) implies reducing the list of potential eligible edges to just a few of them, e.g.: assuming $s = 100$, if we choose $\alpha = 0.35$ then the list of potentially eligible edges will be basically reduced to approximately 5 edges, since $P(X \leq 5) \approx 0.9$. Once a value for α is chosen, the first edge must be selected, this same value 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 $(0.05, 0.25)$ or a trimmed normal with $\mu = 0.15$ and $\sigma = 0.05$ (in this second case, any randomly generated value outside the aforementioned interval is omitted). This way, we not only avoid a fine-tuning processes, 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.

5. Pseudo-code

Fig. 2 shows the main procedure, which drives the solving methodology. First, inputs are entered in the program (line 1) and both nodes and constraints are identified (lines 2 and 3). Then, an efficiency list is constructed (line 4). 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 and Wright Savings heuristic (line 5). 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 (lines 6–10) to generate hundred/thousands of solutions, each of them generated by using our randomized version of the CWS heuristic (line 7). Notice that 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 (line 8). The hash table always keeps the best-known way of connecting a given set of nodes inside a route. In some sense, this could be considered a memory-based 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.

As explained before, the key step in the main procedure is the generation of several hundreds of feasible solutions by using a randomized CWS heuristic. Details of this constructive process can be found at the pseudo-code included in Fig. 3. First, following the original CWS heuristic, a trivial initial solution is built by assigning a different vehicle to each demanding node (line 1). Then, the edges' list is sorted in descending savings order (line 2), and a constructive process starts (lines 3–9). In this process, new edges are randomly selected from the sorted list and their corresponding routes are merged if possible—i.e., if no constraint is violated. As a result, at the end of each constructive process, a random feasible solution is obtained. Of course, the quality of the random solution will depend upon the way in which edges are selected from the list. Fig. 4 shows in detail how a bias is introduced in the selection process by means of a quasi-geometric distribution. First, a random value for beta is selected (line 1) – as explained before, this is usually done by using a uniform distribution. Then, a random number is generated (line 2). Next, for each edge in the sorted list, both the beta value and its order are used to assign its probability of being selected according to a geometric distribution (line 5). Now, if the random number is lower than the cumulative

```
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    end if
9  end while
10 return sol;
end
```

Fig. 3. Construction process.

```
procedure selectEdgeAtRandom(list)
1  beta = generateRandomNumber(a, b); // e.g.: a = 0.05 and b = 0.25
2  randomValue = generateRandomNumber(0, 1);
3  n = 0; cumulativeProbability = 0.0;
4  for each edge in list →
5    edgeProbability = beta * (1 - beta)^n; // geometr. distribution
6    cumulativeProbability += edgeProbability;
7    if randomValue < cumulativeProbability →
8       return edge;
9    else
10      n++;
11   end if
12 end for
13 k = generateIntegerRandomNumber(0, list size);
14 return getEdgeAtPosition(k);
end
```

Fig. 4. Random edge-selection.

probability associated with all processed edges, the last edge processed is selected (line 8); otherwise, another edge from the list is considered and the process starts over. If the list contains only a few number of edges, odds are that none of them will be selected during the iterative process (lines 4–12). In that case, an edge is selected at random – using a uniform distribution – from the list (line 13).

6. Experimental tests

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 [34], 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 50 classical CVRP benchmark instances were selected from the web http://www.branchandcut.org, a reference site which contains detailed information regarding a large number of benchmark instances. Also, a recent paper from Oppen and Lokketangen [43] updating best-know-values for some instances was used as a complementary reference. The selection process was based on the following criteria: (a) only instances for which an 'optimal' solution was given were considered as potentially eligible (this included instances from sets A, B, E, M and P); and (b) only instances offering complete information, e.g., specific routes in 'the optimal' solution, were considered, since we wanted to verify the correctness of the given solution costs. The selected benchmark files are shown on Tables 1–4, which will be commented later. These benchmarks differ in the number of nodes (ranging from 19 to 121), the vehicle capacity, the location of the depot with respect to the clients and their topology.

For instance, Fig. 5 shows the best solution found so far by using our methodology for the E-n51-k5.vrp file, where the depot is at the center. Analogously, Fig. 6 shows the best solution found so far for the A-n80-k10.vrp file. This time, the depot is located at one corner of the scatter plot defined by the set of nodes. Finally, Fig. 7 shows the best solution found so far for the B-n57-k9.vrp file, which is characterized by having a cluster topology.

Table 1
Comparison of results for instances 1–14 (from set A).

a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p
#	Instance	# nodes	Vehicle Capacity	CWS {1}	Gap {1}–{3}	BKS Integer {2}	# routes	Gap {2}–{3}	BKS real {3}	Gap {3}–{4}	OBS {4}	Time (s)	Gap {3}–{5}	Alternative to OBS {5}	Time (s)
1	A-n32-k5	32	100	843.69	7.09%	784	5	−0.48%	787.81	−0.09%	787.08	6	–	–	–
2	A-n33-k5	33	100	712.05	7.44%	661	5	−0.27%	662.76	−0.10%	662.11	2	–	–	–
3	A-n33-k6	33	100	776.26	4.50%	742	6	−0.11%	742.83	−0.02%	742.69	3	–	–	–
4	A-n37-k5	37	100	707.81	5.24%	669	5	−0.53%	672.59	−0.02%	672.47	8	–	–	–
5	A-n38-k5	38	100	768.14	4.63%	730	5	−0.57%	734.18	−0.03%	733.95	7	–	–	–
6	A-n39-k6	39	100	863.08	3.59%	831	6	−0.26%	833.20	0.00%	833.20	1	–	–	–
7	A-n45-k6	45	100	1006.45	6.52%	944	6	−0.09%	944.88	0.00%	944.88	31	–	–	–
8	A-n45-k7	45	100	1199.98	4.59%	1146	7	−0.11%	1147.28	−0.04%	1146.77	45	–	–	–
9	A-n55-k9	55	100	1099.84	2.36%	1073	9	−0.14%	1074.46	0.00%	1074.46	2158	0.05%	1074.96	3
10	A-n60-k9	60	100	1421.88	4.87%	1354	9	−0.13%	1355.80	0.00%	1355.80	38	–	–	–
11	A-n61-k9	61	100	1102.23	6.08%	1034	9	−0.49%	1039.08	0.00%	1039.08	47	–	–	–
12	A-n63-k9	63	100	1687.96	4.06%	1616	9	−0.38%	1622.14	0.00%	1622.14	145	–	–	–
13	A-n65-k9	65	100	1239.42	4.89%	1174	9	−0.65%	1181.69	0.00%	1181.69	30	–	–	–
14	A-n80-k10	80	100	1860.94	5.35%	1763	10	−0.20%	1766.50	0.00%	1766.50	6877	0.12%	1768.70	101

Table 2
Comparison of results for instances 15–27 (from set B).

a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p
#	Instance	# nodes	Vehicle Capacity	CWS {1}	Gap {1}–{3}	BKS Integer {2}	# routes	Gap {2}–{3}	BKS Real {3}	Gap {3}–{4}	OBS {4}	Time (s)	Gap {3}–{5}	Alternative to OBS {5}	Time (s)
15	B-n31-k5	31	100	681.16	0.65%	672	5	−0.70%	676.76	−0.10%	676.09	1	–	–	–
16	B-n35-k5	35	100	978.33	2.30%	955	5	−0.14%	956.29	0.00%	956.29	218	–	–	–
17	B-n39-k5	39	100	566.71	2.43%	549	5	−0.77%	553.27	−0.02%	553.16	17	–	–	–
18	B-n41-k6	41	100	898.09	7.56%	829	6	−0.71%	834.96	0.00%	834.92	2	–	–	–
19	B-n45-k5	45	100	757.16	0.23%	751	5	−0.59%	755.43	−0.16%	754.22	20	–	–	–
20	B-n50-k7	50	100	748.80	0.54%	741	7	−0.51%	744.78	−0.07%	744.23	2	–	–	–
21	B-n52-k7	52	100	764.90	1.98%	747	7	−0.41%	750.08	−0.02%	749.96	40	–	–	–
22	B-n56-k7	56	100	733.74	2.92%	707	7	−0.83%	712.92	0.00%	712.92	1348	–	–	–
23	B-n57-k9	57	100	1653.42	3.10%	1598	9	−0.35%	1603.63	−0.08%	1602.28	3	–	–	–
24	B-n64-k9	64	100	921.56	6.01%	861	9	−0.96%	869.32	−0.12%	868.31	84	–	–	–
25	B-n67-k10	67	100	1099.95	5.83%	1032	10	−0.71%	1039.36	0.00%	1039.36	170	0.01%	1039.46	14
26	B-n68-k9	68	100	1317.77	3.09%	1272	9	−0.49%	1278.21	−0.16%	1276.20	1647	0.00%	1278.23	746
27	B-n78-k10	78	100	1264.56	2.87%	1221	10	−0.67%	1229.27	−0.11%	1227.90	3525	−0.09%	1228.14	2,041

Table 3
Comparison of results for instances 28–36 (from sets E and M).

a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p
#	Instance	# nodes	Vehicle Capacity	CWS {1}	Gap {1}–{3}	BKS Integer {2}	# routes	Gap {2}–{3}	BKS Real {3}	Gap {3}–{4}	OBS {4}	Time (s)	Gap {3}–{5}	Alternative to OBS {5}	Time (s)
28	E-n22-k4	22	6000	388.77	3.59%	375	4	−0.07%	375.28	0.00%	375.28	0	–	–	–
29	E-n30-k3	30	4500	534.45	-0.25%	534	3	−0.34%	535.80	−5.75%	505.01	4	–	–	–
30	E-n33-k4	33	8000	843.10	0.52%	835	4	−0.44%	838.72	−0.13%	837.67	7	–	–	–
31	E-n51-k5	51	160	584.64	11.37%	521	5	−0.75%	524.94	−0.06%	524.61	32	–	–	–
32	E-n76-k7	76	220	737.74	7.29%	682	7	−0.81%	687.60	0.00%	687.60	887	–	–	–
33	E-n76-k10	76	140	900.26	7.51%	830	10	−0.88%	837.36	−0.25%	835.26	21,707	0.14%	838.55	129
34	E-n76-k14	76	100	1073.43	4.55%	1021	14	−0.56%	1026.71	−0.22%	1024.40	4112	0.08%	1027.55	1853
35	M-n101-k10	101	200	833.51	1.67%	820	10	0.02%	819.81	−0.03%	819.56	338	–	–	–
36	M-n121-k7	121	200	1068.14	2.20%	1034	7	−1.07%	1045.16	−0.12%	1043.88	74,488	−0.02%	1044.91	389

Table 4
Comparison of results for instances 37–50 (from set P).

a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p
#	Instance	# nodes	Vehicle Capacity	CWS {1}	Gap {1}–{3}	BKS Integer {2}	# routes	Gap {2}–{3}	BKS Real {3}	Gap {3}–{4}	OBS {4}	Time (s)	Gap {3}–{5}	Alternative to OBS {5}	Time (s)
37	P-n19-k2	19	160	237.90	11.87%	212	2	−0.31%	212.66	0.00%	212.66	16	–	–	–
38	P-n20-k2	20	160	234.00	7.63%	216	2	−0.65%	217.42	0.00%	217.42	41	–	–	–
39	P-n22-k2	22	160	239.50	9.94%	216	2	−0.85%	217.85	0.00%	217.85	9	–	–	–
40	P-n22-k8	22	3000	590.62	-1.80%	603	8	0.26%	601.42	−2.10%	588.79	0	–	–	–
41	P-n40-k5	40	140	518.37	12.27%	458	5	−0.81%	461.73	0.00%	461.73	3	–	–	–
42	P-n50-k8	50	120	674.34	6.22%	631	8	−0.61%	634.85	−0.14%	633.96	14	–	–	–
43	P-n50-k10	50	100	734.32	4.97%	696	10	−0.51%	699.56	0.00%	699.56	379	0.16%	700.66	1
44	P-n51-k10	51	80	790.97	6.53%	741	10	−0.20%	742.48	−0.13%	741.50	19	–	–	–
45	P-n60-k10	60	120	800.20	6.84%	744	10	−0.66%	748.94	−0.12%	748.07	166	–	–	–
46	P-n65-k10	65	130	851.67	6.90%	792	10	−0.59%	796.67	−0.13%	795.66	67	–	–	–
47	P-n70-k10	70	135	896.86	8.05%	827	10	−0.36%	830.02	−0.01%	829.93	994	0.22%	831.81	52
48	P-n76-k4	76	350	689.13	15.20%	593	4	−0.87%	598.22	−0.01%	598.19	704	0.20%	599.44	408
49	P-n76-k5	76	280	698.51	9.99%	627	5	−1.27%	635.04	−0.27%	633.32	73	–	–	–
50	P-n101-k4	101	400	765.38	10.56%	681	4	−1.63%	692.28	−0.03%	692.04	26,465	0.05%	692.64	380

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 Tables 1–4, which contain the following information for each instance: (a) number of instance; (b) name of instance; (c) number of nodes; (d) vehicle capacity; (e) costs associated with the solution given by the parallel version of the CWS heuristic, {1}; (f) gap, expressed as a percentage value, between {1} and {3}, with {3} being the real value associated with the best-know solution; (g) best-known solution and 'optimal' value, {2}, according to [43]; (h) number of routes of the best-known solution; (i) gap between {2} and {3}; (j) verified real costs for the best-know solution, {3}; (k) gap between {3} and {4}, with {4} being the best value obtained by using our approach; (l) our best solution so far, {4}; (m) computational time, in seconds, employed in obtaining {4}; (n) gap between {3} and {5}; (o) alternative solution to {4} obtained with our approach, {5} – these values are only provided if computational times in {4} are considered to be excessive; and (p) computational time employed in obtaining {5}.

First of all, it must be said that some costs values considered in some papers as 'the optimal ones' – i.e., those given in {2} – are true optimal values only when integer-rounded numbers are considered, i.e.: according to our verification process, the 'optimal' costs have been obtained after rounding the initial costs given by the real Euclidean distances between any two nodes. This verification process is based on the detailed routes given for each 'optimal' solution, and makes use of an Excel/VBA application that checks for the feasibility of the proposed solution and also calculates its

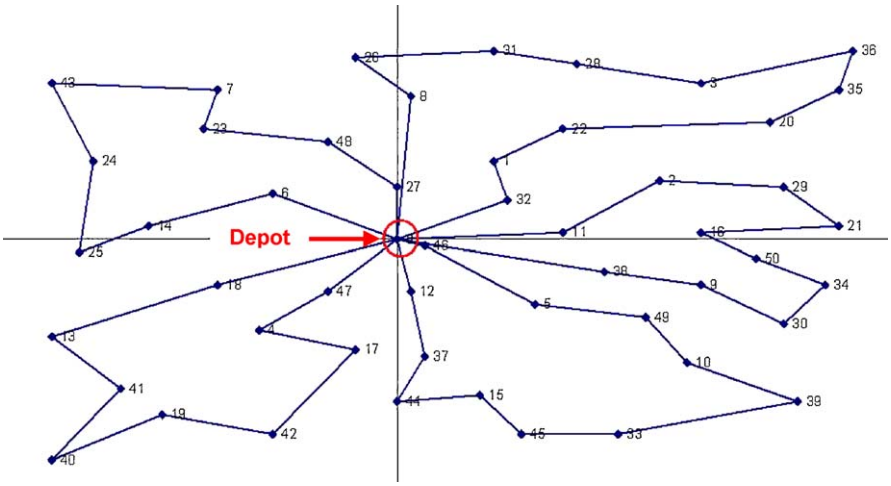

Fig. 5. Best solution found so far for the E-n51-k5.vrp (depot at the center).

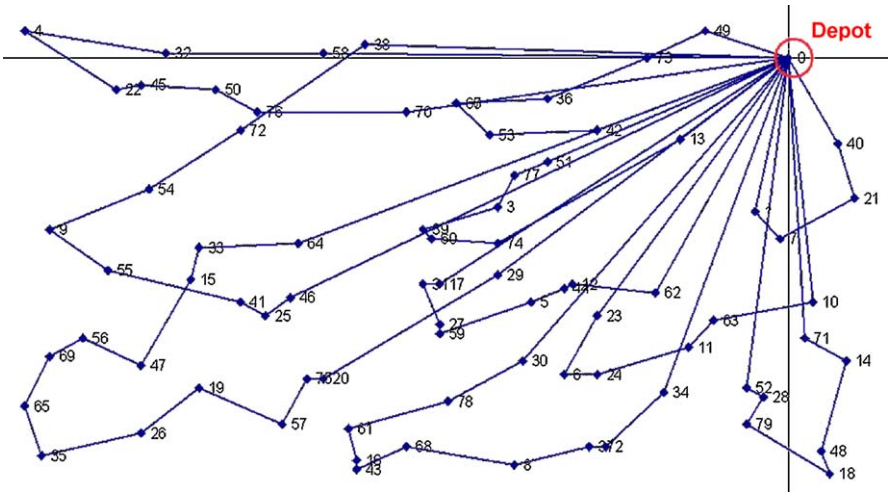

Fig. 6. Best solution found so far for the A-n80-k10.vrp (depot at one corner).

associated costs, both with and without rounding process (Fig. 8). As a result of this verification process, it has been discovered that some 'optimal' solutions are not really optimal when real costs are considered.

This 'rounding factor' explains much of the discrepancies found between {2} and {3}. As far as we are concerned, {3} are the real values to be considered, since {2} accumulate rounding errors. Notice that any positive gap implies that solution costs are higher

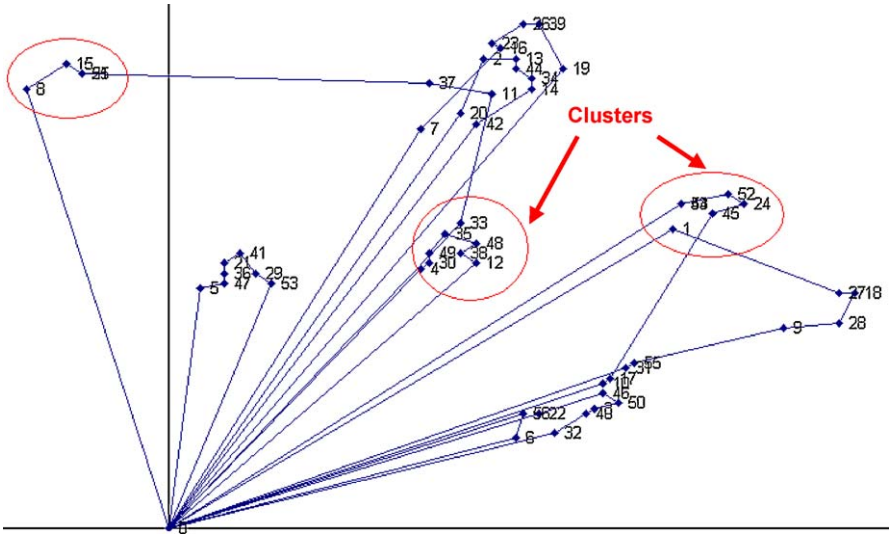

Fig. 7. Best solution found so far for the B-n57-k9.vrp (cluster topology).

	A	D	E	F	G	H	I	J	K	L	M	N	O
1	Solution	<----------		Route Info	---------->		CVRP Solutions Draw & Check						
2	0	Route	Demand	<--------- Costs	--------->		These Excel 2003 macros draw a given CVRP sol and check its feasibility						
3	1	1	14.00	9.43			© Angel A. Juan - ajuanp(@)gmail.com - 080827						
4	52	1	23.00	2.24									
12	0	1	100.00	6.32	28.66		Total # of routes in solution =				10		
13	2	2	17.00	18.97			Total demand to be served =				937.00		
14	63	2	20.00	8.49			Total demand served in this solution =				937.00	OK	
15	49	2	12.00	1.00			Total costs given by Java =				1,229.27	<- User input #1	
16	27	2	23.00	4.47			Total DOUBLE costs of this solution =				1,229.27	OK	
17	56	2	9.00	20.40			Total INT (TRUNC) costs of this solution =				1200	<- Uses trucation	
18	38	2	14.00	6.32			Total INT (ROUND) costs of this solution =				1221	<- Uses rounding	
19	58	2	3.00	1.41			Vehicle capacity =				100.00	<- User input #2	
20	0	2	98.00	8.54	69.61		Max. demand in any route =				100.00	OK	
21	5	3	19.00	29.97			Max. Length (cost) allowed for a route =				1.00E+06	<- User input #3	
22	30	3	4.00	3.61			Max. cost in any route =				223.86	OK	
23	70	3	23.00	3.16									
24	29	3	21.00	3.16			Real costs						
25	37	3	14.00	3.61			Integer costs				Draw Routes	Verify Feasibility	
26	65	3	13.00	7.28									
27	0	3	94.00	28.30	79.08								

Fig. 8. Excel/VBA application used in the verification process of a given 'optimal' solution.

than the ones associated with the verified real costs for the best-known solution or {3}, while a negative gap will imply just the opposite. In the case of the 50 instances considered in this paper, the average gap between {2} and {3} is a negative one of 0.52%. Of course, this makes possible for us to find new solutions outperforming the considered as 'optimal' ones, at least for some of the analyzed instances. Before going any further in this matter, it is worthy to point out that, for the 50 considered instances, the average gap between the CWS solution, {1}, and the real best-know-solution, {3} is about 5.29%. Notice, however, that there are two instances, e.g., E-n30-k3 and P-n22-k8, that show a negative gap, meaning that in those cases the CWS solution outperforms the one considered as 'optimal' by some papers when this solution is calculated in real costs instead of integer (rounded) costs.

Regarding the solutions obtained by using our approach, it should be highlighted that 31-out-of-50 instances offer a negative gap – i.e., they outperform the real best-known solutions – while the remaining 19 instances offer a null gap. In other worlds, our methodology has been able to either match or improve the values in {3} in all 50 instances, showing a negative average gap of 0.21%. Finally, in some cases these values were obtained in just a few seconds of computation. We have to consider that we have performed these experiments in a standard desktop computer and by implementing the algorithm in an interpreted language such as Java. Therefore, it seems reasonable to think that these computational times can be significantly reduced even further with the help of workstations and compiled languages such as C/C++.

7. Discussion of results

As described before, our approach makes use of an iterative process to generate a set of random feasible solutions. According to the experimental tests that we have carried out in the previous section, each algorithm iteration is completed in just a few milliseconds by using a standard computer. By construction, odds

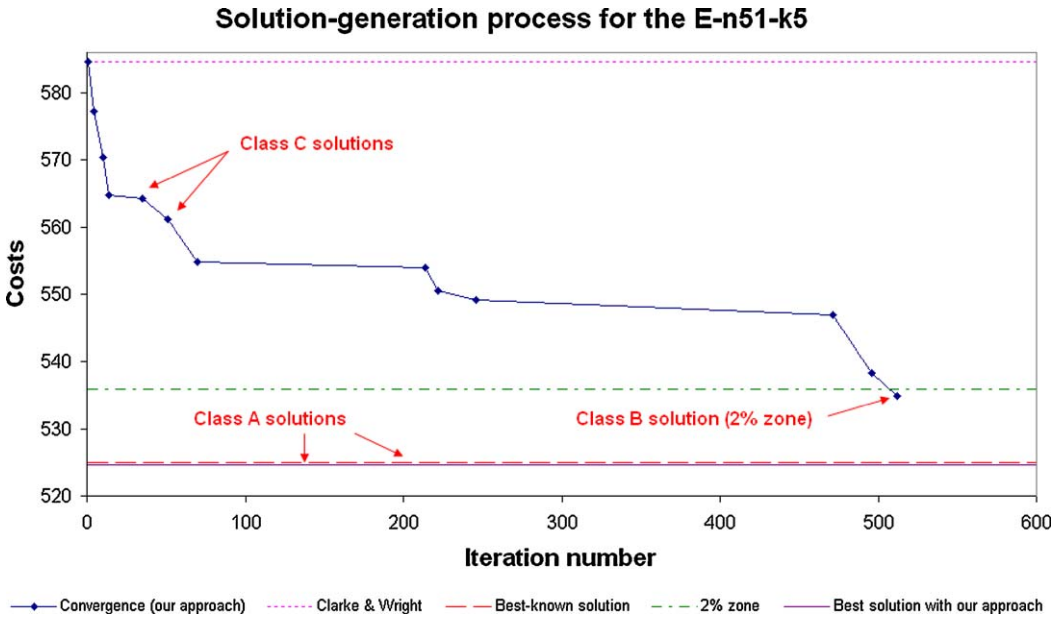

Fig. 9. Different classes of solutions provided by our algorithm.

are that the generated solution outperforms the one given by the CWS heuristic. This means that our approach provides, in almost real-time, what we call "a class C solution", i.e., a feasible solution which outperforms the CWS heuristic (Fig. 9). Moreover, as verified by testing, hundreds 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 fulfills his or her preferences according to his or her utility function.

Furthermore, 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 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 solution for every tested instance.

Another important point to consider here is the simplicity of the presented methodology. As a matter of fact, 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 [27] and [46] some of the most efficient heuristics and meta-heuristics are not used in practice due to the considerable effort they require during the implementation and fine-tuning processes, especially when considering real-life scenarios with an important number of additional constraints. 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.

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, e.g. [36] and, in general, for other combinatorial optimization problems, e.g. [55]. In our opinion, this opens a nice range of potential applications that could be explored in future works.

8. Conclusions

In this paper the SR-GCWS 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 and Wright heuristic with Monte Carlo simulation using a quasi-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 exact and heuristic approaches, 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. According to the tests that we have carried out, which include some scenarios of different dimensions, our methodology has always been able to provide a competitive solution in almost real-time. Finally, because of its simplicity and flexibility, we think that this methodology can easily be adapted to other variants of the vehicle routing problem and even to other combinatorial problems.

Acknowledgements

This work has been partially financed by the Spanish Ministry of Education and Science under grants TRA2006-10639, IAP-020100-2008-11 and DPI2008-03511. We would also like to thank NVIDIA Inc. for their support to our research.

References

[1] E. Alba, B. Dorronsoro, Solving the vehicle routing problem by using cellular genetic algorithms, in: Proceedings of Evolutionary Computation in Combinatorial Optimization – EvoCOP 2004, LNCS 3004, Springer, Berlin, 2004, pp. 11–20.
[2] R. Baldacci, N. Christofides, A. Mingozzi, An exact algorithm for the vehicle routing problem based on the set partitioning formulation with additional cuts, Mathematical Programming Series A 115 (2) (2008) 351–385.
[3] R. Baldacci, A. Mingozzi, A unified exact method for solving different classes of vehicle routing problems, Mathematical Programming Series A 120 (2) (2009) 347–380.
[4] M. Battarra, B. Golden, D. Vigo, Tuning a parametric Clarke–Wright heuristic via a genetic algorithm, Journal of the Operational Research Society 59 (2008) 1568–1572.
[5] J. Beasley, Adapting the savings algorithm for varying inter-customer travel times, Omega 9 (1981) 658–659.
[6] J. Berger, M. Barkaoui, A hybrid genetic algorithm for the capacitated vehicle routing problem, in: Proceedings of the International Genetic and Evolutionary Computation Conference – GECCO03, LNCS 2723, 2003, pp. 646–656.
[7] G.M. Buxey, The vehicle scheduling problem and Monte Carlo simulation, Journal of Operational Research Society 30 (1979) 563–573.
[8] G. Clarke, J. Wright, Scheduling of vehicles from a central depot to a number of delivering points, Operations Research 12 (1964) 568–581.
[9] J.F. Cordeau, M. Gendreau, A. Hertz, G. Laporte, J.S. Sormany, New heuristics for the vehicle routing problem in logistics systems: design and optimization, Kluwer Academic Publishers, 2004.
[10] M. Desrochers, J. Desrosiers, M. Solomon, A new optimization algorithms for the vehicle routing problem with time windows, Operations Research 40 (2) (1992) 342–354.
[11] M. Dror, P. Trudeau, Stochastic vehicle routing with modified savings algorithm, European Journal of Operational Research 23 (1986) 228–235.
[12] J. Faulin, A. Juan, The ALGACEA-1 method for the capacitated vehicle routing problem, International Transactions in Operational Research 15 (2008) 1–23.
[13] J. Faulin, M. Gilibert, A. Juan, R. Ruiz, R.X. Vilajosana, SR-1: a simulation-based algorithm for the capacitated vehicle routing problem, in: Proceedings of the Winter Simulation Conference, 2008.
[14] T.A. Feo, M.G. Resende, A probabilistic heuristic for a computationally difficult set covering problem, Operations Research Letters 8 (1989) 67–71.
[15] T.A. Feo, M.G. Resende, Greedy randomized adaptive search procedures, Journal of Global Optimization 6 (1995) 109–133.
[16] P. Fernández de Córdoba, L.M. García Raffi, A. Mayado, J.M. Sanchis, A real delivery problem dealt with Monte Carlo techniques, TOP 8 (2000) 57–71.
[17] R. Fukasawa, H. Longo, J. Lysgaard, M.P. de Aragao, M. Reis, E. Uchoa, R. Werneck, Robust branch-and-cut-and-price for the capacitated vehicle routing problem, Mathematical Programming Series A 106 (3) (2006) 491–511.
[18] T.J. Gaskell, Bases for the vehicle fleet scheduling, Operational Research Quarterly 18 (2006) 281–295.
[19] M. Gendreau, A. Hertz, G. Laporte, A tabu search heuristic for the vehicle routing problem, Management Science 40 (1994) 1276–1290.
[20] M. Gendreau, J. Potvin, O. Bräysy, G. Hasle, A. Løkketangen, Meta-heuristics for the vehicle routing problem and its extensions: a categorized bibliography, in: The Vehicle Routing Problem: Latest Advanced and New Challenges, Springer, Dordrecht, 2008.
[21] B.E. Gillet, L.R. Miller, A heuristic algorithm for the vehicle dispatch problem, Operations Research 22 (1974) 340–349.
[22] B. Golden, S. Raghavan, E. Edward Wasil, The Vehicle Routing Problem: Latest Advances and New Challenges, Springer, Berlin, 2008.
[23] R.A. Holmes, R.G. Parker, A vehicle scheduling procedure based upon savings and a solution perturbation scheme, Operational Research Quarterly 27 (1976) 83–92.
[24] A. Juan, J. Faulin, J. Jorba, S. Grasman, B. Barrios, SR-2: A hybrid intelligent algorithm for the vehicle routing problem, in: Proceedings of the 8th International Conference on Hybrid Intelligent Systems, IEEE Computer Society, 2008, pp. 78–83.
[25] A. Juan, J. Faulin, R. Ruiz, B. Barrios, M. Gilibert, X. Vilajosana, Using oriented random search to provide a set of alternative solutions to the capacitated vehicle routing problem, in: Operations Research and Cyber-Infrastructure, Springer Operations Research/Computer Science Interfaces Series, vol. 47, 2009, pp. 331–346.
[26] B. Kallehauge, Formulations and exact algorithms for the vehicle routing problem with time windows, Computers & Operations Research 35 (7) (2008) 2307–2330.
[27] G. Kant, M. Jacks, C. Aantjes, Coca-Cola enterprises optimizes vehicle routes for efficient product delivery, Interfaces 38 (2008) 40–50.
[28] J. Kytöjoki, T. Nuortio, O. Bräysy, M. Gendreau, An efficient variable neighborhood search heuristic for very large scale vehicle routing problems, Computers and Operations Research 34 (2007) 2743–2757.
[29] G. Laporte, What you should know about the Vehicle Routing Problem, Naval Research Logistics 54 (2007) 811–819.

[30] G. Laporte, Y. Nobert, Exact algorithms for the vehicle routing problem, Annals of Discrete Mathematics 31 (1987) 147–184.
[31] G. Laporte, F. Semet, Classical heuristics for the capacitated VRP, in: The Vehicle Routing Problem, SIAM Monographs on Discrete Mathematics and Applications, 2002, pp. 109–128.
[32] G. Laporte, M. Gendreau, J.Y. Potvin, F. Semet, Classical and modern heuristics for the vehicle routing problem, International Transactions in Operational Research 7 (2000) 285–300.
[33] A. Law, Simulation Modeling & Analysis, McGraw-Hill, 2007.
[34] P. L'Ecuyer, SSJ: A framework for stochastic simulation in Java, in: Proceedings of the Winter Simulation Conference, 2002, pp. 234–242.
[35] P. L'Ecuyer, Random Number Generation in Simulation, Elsevier Science, Amsterdam, 2006, pp. 55–81.
[36] S. Liang, A. Zincir-Heywood, M. Heywood, Adding more intelligence to the network routing problem: AntNet and GA-agents, Applied Soft Computing 6 (2006) 244–257.
[37] D. Mester, O. Bräysy, Active guided evolution strategies for the large scale vehicle routing problems with time windows, Computers and Operations Research 32 (2005) 1165–1179.
[38] D. Mester, O. Bräysy, Active-guided evolution strategies for the large-scale capacitated vehicle routing problems, Computers and Operations Research 34 (2007) 2964–2975.
[39] MIT, Web site explaining the parallel version of the Clarke & Wright Savings heuristic, http://web.mit.edu/urban_or_book/www/book/chapter6/6.4.12.html (accessed 27.04.09).
[40] R.G. Mole, S.R. Jameson, A sequential route-building algorithm employing a generalised savings criterion, Operational Research Quarterly 27 (1976) 503–511.
[41] D. Naddef, G. Rinaldi, Branch and cut algorithms for the capacitated VRP, in: The vehicle routing problem, SIAM Monographs on Discrete Mathematics and Applications, 2002, pp. 53–81.
[42] Y. Nagata, Edge assembly crossover for the capacitated vehicle routing problem, Lecture Notes in Computer Science, 4446, Springer, Berlin, 2007.
[43] J. Oppen, A. Lokketangen, Arc routing in a node routing environment, Computers and Operations Research 33 (2006) 1033–1055.
[44] H. Paessens, The savings algorithm for the vehicle routing problem, European Journal of Operational Research 34 (1988) 336–344.
[45] D. Pisinger, S. Ropke, A general heuristic for vehicle routing problems, Computers and Operations Research 34 (2007) 2403–2435.
[46] A. Poot, G. Kant, A. Wagelmans, A savings based method for real-life vehicle routing problems, Journal of the Operational Research Society 53 (2002) 57–68.
[47] C. Prins, A simple and effective evolutionary algorithm for the vehicle routing problem, Computers and Operations Research 31 (2004) 1985–2002.
[48] F. Renaud, F.F. Boctor, G. Laporte, An improved petal heuristic for the vehicle routing problem, Journal of the Operational Research Society 47 (1996) 329–336.
[49] D.M. Ryan, C. Hjorring, F. Glover, Extensions of the petal method for vehicle routing, Journal of the Operational Research Society 44 (1993) 289–296.
[50] E. Taillard, Parallel iterative search methods for vehicle routing problems, Networks 23 (1993) 661–673.
[51] C.D. Tarantilis, C.T. Kiranoudis, Boneroute: an adaptative memory-based method for effective fleet management, Annals of Operations Research 115 (2002) 227–241.
[52] P. Toth, D. Vigo, Exact Solution of the Vehicle Routing Problem in Fleet Management and Logistics, Kluwer, Boston, 1998, pp. 1–31.
[53] P. Toth, D. Vigo, The Vehicle Routing Problem, SIAM Monographs on Discrete Mathematics and Applications, 2002.
[54] P. Toth, D. Vigo, The granular tabu search and its application to the vehicle routing problem, INFORMS Journal on Computing 15 (2003) 333–346.
[55] L. Xing, Y. Chen, K.K. Yang, Multi-objective flexible job shop schedule: design and evaluation by simulation modeling, Applied Soft Computing 9 (1) (2009) 362–376.