SORT 38 (1) January-June 2014, 3-12

Improving parametric Clarke and Wright algorithms by means of iterative empirically adjusted greedy heuristics*

Albert Corominas[1], Alberto García-Villoria[1,*] and Rafael Pastor[1]

Abstract

Since Clarke and Wright proposed their well-known savings algorithm for solving the Capacitated Vehicle Routing Problem, several enhancements to the original savings formula have been recently proposed, in the form of parameterisations. In this paper we first propose to use Empirically Adjusted Greedy Heuristics to run these parameterized heuristics and we also consider the addition of new parameters. This approach is shown to improve the savings algorithms proposed in the literature. Moreover, we propose a new procedure which leads to even better solutions, based on what we call Iterative Empirically Adjusted Greedy Heuristics.

MSC: 90C27 (Combinatorial Optimisation).

Keywords: EAGH-1, greedy heuristics, Clarke and Wright savings algorithm, CVRP.

1. Introduction

The Capacitated Vehicle Routing Problem (CVRP) is a well known variant of the NP-hard Vehicle Routing Problem (VRP). Heuristic methods have been proposed to solve it. Among them, the Clarke and Wright savings heuristic (*CW*) (Clarke and Wright, 1964) is one of the most popular: *CW* is simple, easy to implement, very fast and obtains quite good solutions (Altinel and Öncan, 2005).

* Supported by the Spanish MICINN project ENE2010-15509 and co-financed by ERDF

* *Corresponding author:* Alberto García-Villoria, e-mail: alberto.garcia-villoria@upc.edu

[1] Institute of Industrial and Control Engineering (IOC), Av. Diagonal, 647 (Edif. ETSEIB), 11th floor, 08028 Barcelona, Spain. E-mail: {albert.corominas/alberto.garcia-villoria/rafael.pastor}@upc.edu

Received: October 2012
Accepted: February 2014

CW is based on merging tours according to a *savings formula*, which refers to the distance or cost saved by the merged route instead of having the two original ones. Since *CW* was introduced, several enhancements to the original savings formula have been proposed (Gaskell, 1967; Yellow, 1970; Paessens, 1988; and recently Altinel and Öncan, 2005; Doyuran and Çatay, 2011). The best results achieved with a savings heuristic have been obtained by the Doyran and Çatay's *CW* version (let it be called *CW-DC*) and by the Altinel and Öncan's *CW* version (let it be called *CW-AO*).

All aforementioned enhanced savings formulae are parameterized. The performance of these parametric savings heuristics depends on the parameter values. A common practice in the literature is to set the parameter values and then run the algorithms using these values, which remain fixed during the run (Adenso-Díaz and Laguna, 2006). This approach is followed by Battarra et al. (2008) and Corominas et al. (2010) to set the parameter values of *CW-AO*. However, this approach has the drawback of needing a sufficiently representative set of training instances to calibrate the algorithm so the values of the parameters are suitable for any instance of the problem. On the other hand, to solve an instance, Altinel and Öncan (2005), and Doyran and Çatay (2011) run *CW-AO* and *CW-DC*, respectively, with a fixed set of 8,820 different combinations of parameter values and select the best combination. This other approach has the advantage that the parameters of the heuristics are specifically tuned for each instance and the drawback that more computing time is needed to solve each instance.

In this paper we follow and focus on the second approach. The objective is to improve the results obtained in the literature, by means of: i) considering to add new parameters, ii) using a more sophisticated procedure in the literature to obtain suitable parameter values according to the particular instance to be solved, which is known as EAGH-1 (Empirically Adjusted Greedy Heuristics-1) (Corominas, 2005), and iii) designing a new procedure based on EAGH-1 that we call Iterative EAGH-1 (IEAGH-1). IEAGH-1 always ensures solutions better than or equal to the solutions obtained by EAGH-1.

The remainder of the paper is organised as follows. Section 2 defines the CVRP and describes the parametric algorithms based on *CW* proposed in the literature. Section 3 explains EAGH-1 and proposes the new IEAGH-1 procedure. Section 4 presents the results of a computational experiment. Finally, the conclusions are given in Section 5.

2. Parametric Clarke and Wright algorithms to solve the CVRP

The VRP consists in finding the set of routes that minimises the total routing cost. The routes are designed for a fleet of vehicles that has to serve a set of customers with positive demand from one or several depots, subject to various constraints. The CVRP is a VRP variant in which all the vehicles have the same capacity, C, and there is only one depot. The formulation of the CVRP is as follows. Let $G = (V, E)$ be an undirected graph. V is the set of nodes $\{0, 1, \ldots, n\}$, in which node 0 represents the depot and the other n nodes represent the customers, and E is the set of edges. Each vehicle has a

capacity C and each customer i has a demand $0 < d_i \leq C$ $(i = 1,\ldots,n)$. For each edge $(i,j) \in E$ $(i = 0,\ldots,n; j = 0,\ldots,n)$, a traversing cost c_{ij} is associated. The objective is to find the routes that minimise the total cost so that each customer is visited only once and each route starts and ends at the depot.

Clarke and Wright (1964) proposed a greedy heuristic (*CW*) which solves the CVRP. Initially, this heuristic considers n routes to visit all customers, where each route includes only one customer. Next, at each iteration the two routes that can be feasibly merged with the largest *saving* are chosen to be merged. The saving $s_{ij} = c_{i0} + c_{0j} - c_{ij}$ is the cost saved by the merge of the routes $(0,\ldots,i,0)$ and $(0,j,\ldots,0)$.

Based on CW saving formula, Gaskell (1967) and Yellow (1970) proposed the parameterised saving expression $s_{ij} = c_{i0} + c_{0j} - \lambda c_{ij}$, where λ is a parameter that weights the relative importance of c_{ij}. Paessens (1988) proposed $s_{ij} = c_{i0} + c_{0j} - \lambda c_{ij} + \mu\,|c_{0i} - c_{j0}|$, where μ is another weight parameter. Later, Altinel and Öncan (2005) included the customer demands in their *CW* enhancement (*CW-AO*): $s_{ij} = c_{i0} + c_{0j} - \lambda c_{ij} + \mu\,|c_{0i} - c_{j0}| + \upsilon\frac{d_i+d_j}{\bar{d}}$, where $\bar{d}$ is the average demand and υ is a new weight parameter. To set the specific values of the three parameters when solving every instance, the authors vary the values of λ between 0.1 and 2, and the values of μ and υ between 0 and 2, using a step size equal to 0.1. So, *CW-AO* is executed 8,820 times with all combinations of the parameter values and the best solution obtained is chosen. This method improves on the original *CW* but requires much more computing time.

Lately, Doyuran and Çatay (2011) proposed the following savings formula:

$$s_{ij} = \frac{c_{i0} + c_{0j} - \lambda c_{ij}}{c^{\max}} + \mu\frac{\cos\theta_{ij}\left|c^{\max} - (c_{0i} - c_{j0})/2\right|}{c^{\max}} + \upsilon\frac{\left|\bar{d} - (d_i + d_j)/2\right|}{d^{\max}},$$

where θ_{ij} is the angular distance between the customers i and j with respect to the depot, c^{max} is the longest distance among all customer pairs and $d^{\max}$ is the maximum demand among all customers. This savings function has the same 3 parameters that the Altinel and Öncan function, and Doyuran and Çatay also propose to run *CW-DC* 8,820 times with the combinations of the parameter values. But the variation of the parameter values differs: λ between 0.1 and 2, μ between 0 and 2, and υ between -1 and 1, using a step size equal to 0.1.

3. EAGH-1 and IEAGH-1 for the CVRP

3.1. EAGH-1

The main characteristic of a greedy heuristic is that at each iteration an irreversible decision is taken according to an index value associated with each possible decision. When two or more indices are considered in a greedy heuristic, infinite mix-indices

can be generated by linear combination and, therefore, an infinite set H of heuristics can be obtained. Regarding savings algorithms, the infinite set of heuristics H can be defined by the following function h which depends on the set of attributes of the decision ($a_{ij} = \{c_{i0}, c_{0j}, c_{ij}, c_{0i}, c_{j0}, d_i, d_j, \bar{d}\}$) and a set of parameters ($\Pi = \{\lambda, \mu, \upsilon\}$) that weight the elementary indices: $h(a_{ij}, \Pi) = c_{i0} + c_{0j} - \lambda c_{ij} + \mu |c_{0i} - c_{j0}| + \upsilon \frac{d_i + d_j}{\bar{d}}$.

The EAGH-1 procedure (Corominas, 2005) seeks the best heuristic function $h \in H$ for a given instance according to the objective function $f(X_\Pi)$ to be minimised, where X_Π is the solution obtained by $h(a_{ij}, \Pi)$. In the case of the CVRP, f is the total route cost. Note that to find the best heuristic function $h(a_{ij}, \Pi)$ is equivalent to find the Π parameter values that minimise f. Because the function f is not expected to have any special or recognisable property, only a direct optimisation algorithm (i.e. an optimisation algorithm that only uses the values of the function) may be used. Corominas (2005) proposes to apply the Nelder and Mead algorithm (N&M), also known as the flexible polyhedron algorithm (Corominas et al., 2010), as the optimisation algorithm for minimising f.

N&M is based on $q+1$ points ($q = |\Pi|$) that are the vertices of a q-dimensional simplex: $V_0, V_1, \ldots, V_i, \ldots, V_q$. The coordinates of the vertices represent the parameter values (Π), and each vertex is evaluated using f, i.e., the value of the solution (according to f) obtained when the instance is solved using $h(a_{ij}, \Pi)$ and Π is defined by the coordinates of the vertex. At each iteration of N&M, the vertices of the simplex are moved over the q-dimensional space according to their evaluations. N&M starts from an initial regular simplex. To build it, we have to provide N&M with an initial vertex (V_0) and the length of the edges of the initial simplex (δ). Thus, the N&M parameters are f, V_0 and δ. A more detailed description of N&M is provided in the Appendix.

We propose to solve the CVRP using four procedures based on EAGH-1, and the *CW-AO* and *CW-DC* saving formulae. First define the following two sets of decision attributes: $a_{ij}^{AO} = \{c_{i0}, c_{0j}, c_{ij}, c_{0i}, c_{j0}, d_i, d_j, \bar{d}\}$ and $a_{ij}^{DC} = \{c_{i0}, c_{0j}, c_{ij}, c_{0i}, c_{j0}, d_i, d_j, \bar{d}, \theta_{ij}, c^{\max}, d^{\max}\}$, and define the following three sets of parameters: $\Pi_3 = \{\lambda, \mu, \upsilon\}$, $\Pi_6 = \{\lambda, \mu, \upsilon, \beta_1, \beta_2, \beta_3\}$ and $\Pi_8 = \{\lambda, \mu, \upsilon, \beta_1, \beta_2, \beta_3, \beta_4, \beta_5\}$. Finally, define the following four sets of heuristics:

$$h_3^{AO}(a_{ij}^{AO}, \Pi_3) = c_{i0} + c_{0j} - \lambda c_{ij} + \mu |c_{0i} - c_{j0}| + \upsilon \frac{d_i + d_j}{\bar{d}}$$

$$h_8^{AO}(a_{ij}^{AO}, \Pi_8) = c_{i0}^{\beta_1} + c_{0j}^{\beta_2} - \lambda c_{ij}^{\beta_3} + \mu |c_{0i} - c_{j0}|^{\beta_4} + \upsilon \left(\frac{d_i + d_j}{\bar{d}}\right)^{\beta_5}$$

$$h_3^{DC}(a_{ij}^{DC}, \Pi_3) = \frac{c_{i0} + c_{0j} - \lambda c_{ij}}{c^{\max}} + \mu \frac{\cos\theta_{ij} \left| c^{\max} - \frac{(c_{0i} - c_{j0})}{2} \right|}{c^{\max}} + \upsilon \frac{\left| \bar{d} - \frac{(d_i + d_j)}{2} \right|}{d^{\max}}$$

$$h_6^{DC}(a_{ij}^{DC}, \Pi_6) = \left(\frac{c_{i0} + c_{0j} - \lambda c_{ij}}{c^{\max}}\right)^{\beta_1} + \mu \left(\frac{\cos\theta_{ij} \left| c^{\max} - \frac{(c_{0i} - c_{j0})}{2} \right|}{c^{\max}} \right)^{\beta_2} + \upsilon \left(\frac{\left| \bar{d} - \frac{(d_i + d_j)}{2} \right|}{d^{\max}} \right)^{\beta_3}$$

The four procedures based on EAGH-1 that we propose, $EAGH-1_3^{AO}$, $EAGH-1_3^{AO}$, $EAGH-1_3^{DC}$ and $EAGH-1_6^{DC}$, consist in applying EAGH-1 to the set of heuristics defined by h_3^{AO}, h_8^{AO}, h_3^{DC} and h_6^{DC}, respectively. Regarding the initial vertex V_0, we use a point that corresponds to the original Clarke and Wright algorithm: $\lambda=1$, $\mu=0$, $\upsilon=0$, $\beta_1=\beta_2=\ldots=1$. Regarding the initial length of the edges δ, since different solutions may be obtained according to its value, 20 δ values between 0.25 and 5 with a step size equal to 0.25 have been considered. Thus, $EAGH-1_3^{AO}$, $EAGH-1_3^{AO}$, $EAGH-1_3^{DC}$ and $EAGH-1_6^{DC}$ are run using the 20 δ values and the best found solution is returned.

Since the sets of heuristics defined by h_3^{AO} and h_3^{DC} are subsets of the sets defined by h_8^{AO} and h_6^{DC}, respectively, we expect that better solutions are found by $EAGH-1_3^{AO}$ and $EAGH-1_6^{DC}$, although we also expected that the computing time will be larger since the best heuristic will be sought in a larger search space.

3.2. IEAGH-1

We propose a new procedure that we call Iterative EAGH-1 (IEAGH-1). It takes advantage of the two following properties of the optimisation algorithm that is used by EAGH-1 (the N&M algorithm): i) it is recommended to use a good starting point V_0, and ii) N&M ensures that the best set of parameters found Π^* is always better than or equal to V_0 (i.e, $f(X_{\Pi^*}) \leq f(X_{V_0})$).

IEAGH-1 is based on applying iteratively EAGH-1. It first apply EAGH-1. Let $\prod_0^*$ be the set of parameters found. Then, at each iteration *it* ($it=1,2,\ldots$), it applies EAGH-1 using the initial vertex $\prod_{it-1}^*$, where $\prod_{it}^*$ is the set of parameters found by EAHG-1 at the iteration *it*. The stop condition of IEAGH-1 is that no improvement is achieved, that is, $f(X_{\Pi_{it-1}^*}) = f(X_{\Pi_{it}^*})$. Note that IEAGH-1 always ensures solutions better than or equal to EAGH-1 but the computing time will be, of course, larger.

Analogously to the EAGH-1 procedures proposed for the CVRP, we propose four IEAGH-1 procedures for solving this problem: $IEAGH-1_3^{AO}$, $IEAGH-1_8^{AO}$, $IEAGH-1_3^{DC}$ and $IEAGH-1_6^{DC}$. First, $EAGH-1_3^{AO}$, $EAGH-1_3^{AO}$, $EAGH-1_3^{DC}$ and $EAGH-1_6^{DC}$ are run, respectively, using the 20 aforementioned δ values. Then, at each, iteration, EAGH-1 is run using only 5 δ values around the δ value that returned the best solution at previous iteration. That is, if at iteration $it-1$ the best solution was found using a δ value equal to δ^*, then the δ values $\delta^*-0.50$, $\delta^*-0.25$, δ^*, $\delta^*+0.25$, $\delta^*+0.50$ are used at the iteration *it*.

Table 1: *Average percentage solution improvement over CW (average computing times, in seconds).*

	Global	**P**	**A**	**B**	**CE**	**CMT**
Altinel and Öncan	2.97 (6.96)	4.47 (3.96)	2.44 (4.69)	2.10 (4.79)	3.26 (7.11)	2.88 (32.15)
Doyuran and Çatay	3.16 (7.43)	5.20 (4.85)	2.47 (5.14)	2.10 (5.24)	3.22 (7.21)	2.86 (31.80)
$EAGH - 1_3^{AO}$	3.22 (5.73)	4.98 (3.74)	2.62 (4.08)	2.21 (4.34)	3.34 (5.65)	3.18 (23.00)
$EAGH - 1_3^{AO}$	3.40 (37.73)	5.39 (24.19)	2.69 (26.90)	2.35 (26.91)	3.43 (37.11)	3.31 (158.31)
$EAGH - 1_3^{DC}$	3.17 (6.53)	5.04 (4.26)	2.54 (4.72)	2.20 (5.10)	3.12 (6.44)	2.97 (25.45)
$EAGH - 1_6^{DC}$	3.46 (24.25)	5.47 (15.42)	2.85 (17.42)	2.33 (18.96)	3.53 (23.58)	3.18 (96.47)
$IEAGH - 1_3^{AO}$	3.26 (12.60)	5.03 (8.30)	2.64 (9.16)	2.27 (9.84)	3.34 (12.16)	3.20 (48.99)
$IEAGH - 1_8^{AO}$	3.51 (88.85)	5.40 (51.15)	2.92 (64.49)	2.39 (64.03)	3.55 (83.34)	3.46 (389.21)
$IEAGH - 1_3^{DC}$	3.29 (22.71)	5.15 (14.04)	2.63 (16.77)	2.35 (18.22)	3.29 (21.56)	3.13 (88.90)
$IEAGH - 1_6^{DC}$	3.51 (30.83)	5.47 (18.71)	2.89 (21.63)	2.42 (24.36)	3.53 (30.18)	3.27 (126.46)

4. Computational experiment

The test instances used in the computational experiment are the same instances used in Altinel and Öncan (2005), which were also used in Doyuran and Çatay (2011). Namely, 72 instances of Augerat et al. (1995) grouped in three sets (22 in set P, 27 in set A and 23 in set B), 8 instances of Christofides and Eilon (1969) (set CE) and 7 instances of Christofides et al. (1979) (set CMT).

The algorithms were coded in Java and the computational experiment was carried out using a 1.17 GHz Intel Core i7 with 3.0 GB of RAM. Because the Altinel and Öncan (2005) and Doyuran and Çatay (2011) experiments were carried out on computers different from ours, we coded and ran again their experiments on our computer so the comparison of the computational times is fair. We found a slight variability in the numerical results when we rerun their experiments. This phenomenon in savings algorithms have been reported, for instance, in Laporte et al. (2000) and in Doyuran and Çatay (2011). The reason that may cause this difference is the computer code (for instance, in savings

algorithms it is not specified how to break the ties between pair of costumers with equal savings). For the sake of consistency, all results reported here are the ones found with our code.

As it is done in the literature, we consider the following two criteria when evaluating the procedures: the average percentage solution improvement over the original *CW* and the average computing time. Table 1 shows the obtained results, in which the grey rows indicate dominated procedures (worse solutions, on average, are obtained in equal or higher computing time or equal solutions, on average, are obtained in higher computing time).

We can see in Table 1 that the results of Altinel and Öncan (2005) and Doyuran and Çatay (2011) are dominated by $EAGH-1_3^{AO}$ and $EAGH-1_3^{DC}$. Moreover, $EAGH-1_3^{DC}$ is dominated by $EAGH-1_3^{AO}$. Therefore, we compare $EAGH-1_3^{AO}$ with them. Specifically, the average improvement obtained with $EAGH-1_3^{AO}$ over *CW* is 7.76%, 1.86% and 1.55% better than the improvement obtained in Altinel and Öncan (2005), Doyuran and Çatay (2011) and by $EAGH-1_3^{DC}$, respectively, and the average computing time is 1.21, 1.30 and 1.14 times smaller, respectively. Other existing dominated procedures are $EAGH-1_3^{AO}$ (by $EAGH-1_6^{DC}$) and $IEAGH-1_8^{AO}$ (by $IEAGH-1_6^{DC}$). On average, the best solutions obtained with EAGH are those that correspond to $EAGH-1_6^{DC}$, whose improvement over *CW* is 14.16%, 8.67% and 6.94% better than the improvement achieved by Altinel and Öncan, Doyuran and Çatay and $EAGH-1_3^{AO}$, respectively.

As we expected, all IEAGH-1 procedures obtain better solutions than their respective EAGH-1 procedures, although the computing times are larger. The average improvements over *CW* obtained with $IEAGH-1_3^{AO}$, $IEAGH-1_8^{AO}$, $IEAGH-1_3^{DC}$ and $IEAGH-1_6^{DC}$ are 1.23%, 3.13%, 3.65% and 1.42% better than the improvements obtained with $EAGH-1_3^{AO}$, $EAGH-1_3^{AO}$, $EAGH-1_3^{DC}$ and $EAGH-1_6^{DC}$, respectively, and the average computing times are 2.20, 2.35, 3.48 and 1.27 times larger, respectively. The best solutions are obtained with $IEAGH-1_6^{DC}$, which dominates $IEAGH-1_8^{AO}$ (they obtain the same average improvement but the computing time of $IEAGH-1_8^{AO}$ is larger). Specifically, the $IEAGH-1_6^{DC}$ improvement over *CW* is 15.38% and 9.97% better than the improvement achieved by Altinel and Öncan, and Doyuran and Çatay, respectively.

5. Conclusions

In this paper we have improved the resolution of the CRVP with parametric Clarke and Wright savings heuristics. To achieve this objective, we have considered the addition of new parameters in the parameterized savings formula.

We first propose to use EAGH-1 (Corominas, 2005) and the computational experiment shows that better solutions, on average, can be found with less computing time. Specifically, the $EAGH-1_3^{AO}$ improvement over *CW* is, on average, 1.86% better than

Doyuran and Çatay's improvement whereas their average computing times are 5.73 s and 7.43 s, respectively. With some more computing time (24.25 s), $EAGH-1_6^{DC}$ is able to obtain an improvement 8.67% better than the Doyuran and Çatay's procedure improvement.

Moreover, we propose a new procedure based on EAGH-1 that we call Iterative EAGH-1 (IEAGH-1). It is shown that the solutions are improved with respect to the ones obtained by EAGH-1 at the expense of a larger computing time. The best results are obtained by $IEAGH-1_6^{DC}$, which slightly improves $EAGH-1_6^{DC}$ but the average computing time is gone up by 6.58 s.

Although we have proposed IEAGH-1 to solve the CRVP, this procedure can be also applied to solve other combinatorial optimisation problems.

APPENDIX

The N&M algorithm is a direct search method for minimising $f(x)$ where $f:\mathbb{R}^q\to\mathbb{R}$ is the objective function and q the dimension. It is based on $q+1$ points that are the vertices of a simplex in the q-dimensional space: $x_1,x_2,\ldots,x_{q+1}$. N&M starts from an initial simplex (usually regular) and iteratively moves the vertices over the q-dimensional space according to their objective function values until the differences between the values of the vertices are small enough and the simplex is small enough.

At each iteration of N&M, the vertices of the simplex are labelled and ordered such that $f(x_1)\leq f(x_2)\leq\ldots\leq f(x_{q+1})$. In the case of ties, the oldest vertex has priority. Let $x_r=\bar{x}+\alpha(\bar{x}-x_{q+1})$ be the reflection of x_{q+1}, where $\bar{x}$ is the centroid of the q best vertices (i.e., $\bar{x}=\sum_{i=1}^{q}x_i/q$) and $\alpha>0$ is a parameter. Four cases are considered according to the $f(x_r)$ value:

1. *Expansion*. If $f(x_r)<f(x_1)$ then calculate $x_e=\bar{x}+\gamma(x_r-\bar{x})$, where $\gamma>1$ is a parameter. If $f(x_e)<f(x_1)$, replace x_{q+1} with x_e; otherwise, replace x_{q+1} with x_r.
2. *Reflection*. If $f(x_1)\leq f(x_r)<f(x_q)$ then replace x_{q+1} with x_r.
3. *Outside contraction*. If $f(x_q)\leq f(x_r)<f(x_{q+1})$ then calculate $x_{oc}=\bar{x}+\eta(x_r-\bar{x})$, where $0<\eta<1$ is a parameter. If $f(x_{oc})<f(x_r)$, replace x_{q+1} with x_{oc}; otherwise, replace x_{q+1} with x_r and shrink all vertices except x_1: $x_i=x_1+\delta(x_i-x_1)$ $i=2,\ldots,q+1$, where $0<\delta<1$ is a parameter.
4. *Inside contraction*. If $f(x_{q+1})\leq f(x_r)$ then calculate $x_{ic}=\bar{x}+\eta(x_{q+1}-\bar{x})$. If $f(x_{ic})<f(x_{q+1})$, replace x_{q+1} with x_{ic}; otherwise, shrink all vertices except x_1 as in 3).

The values of parameters α, γ, η and δ that we have adopted are 1, 2, 0.5 and 0.5, respectively, which have been almost always used in the literature (Lagarias et al., 1998).

For more details of N&M, see Nelder and Mead (1965) or Corominas et al. (2010).

References

Adenso-Díaz, B. and Laguna, M. (2006). Fine-tuning of algorithms using fractional experimental designs and local search. *Operations Research*, 54, 99–114.

Altinel, I. K. and Öncan, T. (2005). A new enhancement of the Clarke and Wright savings heuristic for the capacitated vehicle routing problem. *Journal of the Operational Research Society*, 56, 954–61.

Augerat, P., Belenguer, J. M., Benavent, E., Corberán, A., Naddef, D. and Rinaldi, G. (1995). Computational results with a Branch and Cut Code for the Capacitated Vehicle Routing Problem. *Technical report RR 949-M*, U. Joseph Fourier, France.

Battarra, M., Golden, B. and Vigo, D. (2008). Tuning a parametric Clarke-Wright heuristic via a genetic algorithm. *Journal of the Operational Research Society*, 59, 1568–72.

Christofides, N. and Eilon, S. (1969). An algorithm for the vehicle routing dispatching problem. *Operations Research Quarterly*, 20, 309–18.

Christofides, N., Mingozzi, A. and Toth, P. (1979). The Vehicle Routing Problem. *Combinatorial Optimization*, Christofides, N., Mingozzi, A., Toth, P. and Sandi, C. (Eds.), Wiley, Chichester, 318–38.

Clarke, G. and Wright, J. (1964). Scheduling of vehicles from a central depot to a number of delivery points. *Operations Research*, 12, 568–81.

Corominas, A. (2005). Empirically Adjusted Greedy Algoriths (EAGH): A new approach to solving combinatorial optimisation problems. *Working paper IOC-DT-P-2005-22*, Universitat Politècnica de Catalunya, Spain.

Corominas, A., García-Villoria, A. and Pastor, R. (2010). Fine-tuning a parametric Clarke and Wright heuristic by means of EAGH (empirically adjusted greedy heuristics). *Journal of the Operational Research Society*, 61, 1309–14.

Doyuran, T. and Çatay, B. (2011). A robust enhancement to the Clarke-Wright savings algorithm. *Journal of the Operational Research Society*, 62, 223–31.

Gaskell, T. J. (1967). Bases for vehicle fleet scheduling. *Operations Research Quarterly*, 18, 281–95.

Lagarias, J. C., Reeds, J. A., Wright, M. H. and Wright, P. E. (1998). Convergence properties of the Nelder-Mead simplex method in low dimensions. *SIAM Journal of Optimization*, 9, 112–47.

Laporte, G., Gendreau, M., Potvin, J.and Semet, F. (2000). Classical and modern heuristics for the vehicle routing problem. *International Transactions in Operational Research*, 7, 285–300.

Nelder, J. A. and Mead, R. (1965). A simplex method for function minimization. *The Computer Journal*, 7, 308–13.

Paessens, H. (1988). The savings algorithm for the vehicle routing problem. *European Journal of Operational Research*, 34, 336–44.

Yellow, P. (1970). A computational modification to the savings method of vehicle scheduling. *Operations Research Quarterly*, 21, 281–83.