Available online at www.sciencedirect.com

SCIENCE DIRECT®

Information Processing Letters ••• (••••) •••–•••

Information Processing Letters

www.elsevier.com/locate/ipl

Computing nine new best-so-far solutions for Capacitated VRP with a cellular Genetic Algorithm

Enrique Alba [*], Bernabé Dorronsoro

Department of Computer Science, University of Málaga, Spain

Received 15 April 2004; received in revised form 24 June 2005

Communicated by L. Boasson

Abstract

The Vehicle Routing Problem (VRP) is a hard combinatorial problem with numerous industrial applications. Among the large number of extensions to the canonical VRP, we study the Capacitated VRP (CVRP), which is mainly characterized by using vehicles of the same capacity. A cellular Genetic Algorithm (cGA)—a kind of decentralized population based heuristic—is used for solving CVRP, improving several of the best existing results so far in the literature. Our study shows a high performance in terms of the quality of the solutions found and the number of function evaluations (effort).
© 2006 Elsevier B.V. All rights reserved.

Keywords: Cellular Genetic Algorithm; Hybrid Genetic Algorithm; Vehicle Routing Problem

1. Introduction

This paper is devoted to solve the Capacitated VRP (CVRP), an extension of VRP [1] in which all vehicles have the same capacity. Formally, CVRP is defined on an undirected graph $G = (\vec{V}, \vec{E})$ where $\vec{V} = \{v_0, v_1, \ldots, v_n\}$ is a vertex set and $\vec{E} = \{(v_i, v_j)/v_i, v_j \in \vec{V}, i < j\}$ is an edge set. Vertex v_0 stands for the *depot*, and it is from where m identical vehicles of capacity Q must serve all the *cities* or *customers*, represented by the set of n vertices $\{v_1, \ldots, v_n\}$. We define on E a non-negative *cost*, *distance* or *travel time* matrix $C = (c_{ij})$ between customers v_i and v_j. Each customer v_i has non-negative demand of goods q_i and drop time δ_i (time to unload all goods). Let be $\vec{V}_1, \ldots, \vec{V}_m$ a partition of $\vec{V}$, a route $\vec{R}_i$ is a permutation of the customers in $\vec{V}_i$ specifying the order of visiting them, starting and finishing at the depot v_0. The cost of the problem solution—$F(\vec{S})$—is the sum of the costs of its routes $\vec{R}_i$:

$$F(\vec{S}) = \sum_{i=1}^{m} \mathrm{Cost}(\vec{R}_i);$$
$$\mathrm{Cost}(\vec{R}_i) = \sum_{j=0}^{k} c_{j,j+1} + \sum_{j=0}^{k} \delta_j \quad . \qquad (1)$$

CVRP consists in determining a set of m routes (i) of minimum total cost (see Eq. (1)); (ii) starting and ending at the depot v_0; and such that (iii) each customer is visited exactly once by exactly one vehicle; subject to the restriction that (iv) the total demand of any route does not exceed Q ($\sum_{v_j \in \vec{R}_i} q_j \leqslant Q$). We additionally con-

[*] Corresponding author.
E-mail addresses: eat@lcc.uma.es (E. Alba), dorronsoro@uma.es (B. Dorronsoro).

0020-0190/$ – see front matter © 2006 Elsevier B.V. All rights reserved.
doi:10.1016/j.ipl.2006.02.006

sider that (v) the total duration of any route is not longer than a preset bound D ($\mathrm{Cost}(\vec{R}_i) \leqslant D$).

As we previously did in [2], we face the problem with a cellular Genetic Algorithm (cGA) [3,4] hybridized with a specialized local search method. Cellular GAs are a subclass of GAs in which the population is structured in a specified topology (usually a mesh), and the genetic operations may only take place in a small neighborhood of each individual.

The contribution of this work is to show the accurate behavior of a powerful yet simple cGA for a complex combinatorial problem like CVRP. For that, we compare our results with the best existing ones in the literature, being able of improving them in many cases. In [2] the reader can find a seminal work with a comparison between our cGA, and some other heuristics for a well-known benchmark (Christofides et al. [5]). In that work, we showed the advantages of embedding local search techniques into a cGA for solving CVRP, since our cGA was the best algorithm out of all the compared ones in terms of accuracy and time. Here we focus in providing new optimal values for three additional benchmarks (Van Breedam [6], Golden et al. [7], and Taillard [8]). These new optimal solutions will be a reference for future comparisons for other authors.

The paper is organized as follows. In Section 2, the proposed cGA is described. Section 3 contains our results. Finally, the conclusions and future research lines are given in Section 4.

2. The proposed algorithm

In this section, we present a brief description of our cGA, called JCell2o1i (a detailed description is given in [2]). In JCell2o1i, solutions are represented as permutations of integers. Each permutation will contain both customers and route splitters (delimiting different routes), so we will use a permutation of numbers $[0 \ldots n-1]$ with length $n = c + k$ for representing a solution for the CVRP with c customers and a maximum of $k+1$ possible routes. Customers are represented with numbers $[0 \ldots c-1]$, while route splitters belong to the range $[c \ldots n-1]$. Each route is made of the customers situated between two route splitters in the individual.

In our basic cGA, the population is structured in a 2D toroidal grid, and the neighborhood defined on it—line 5—always contains 5 individuals: the considered one—at position (x, y)—plus the North, East, West, and South individuals (called NEWS). The first parent is chosen by using binary tournament selection (BT) inside this neighborhood (line 6), while the other parent is the current individual itself (line 7). The algorithm iteratively considers as current each individual in the grid. The two parents can be the same individual (replacement) or not.

```
 1: proc Steps_Up(cga) // Algorithm parameters in 'cga'
 2: while not Termination_Condition() do
 3:    for x ← 1 to WIDTH do
 4:       for y ← 1 to HEIGHT do
 5:          n_list ← Get_Neighb(cga, x, y);
 6:          parent1 ← Binary_Tournament(n_list);
 7:          parent2 ← Individual_At(cga, x, y);
 8:          offspring ← ERX(cga. Pc,parent1,parent2);
 9:          offspring ← Combined_Mut(cga.Pm, offspring);
10:          sol_Opt ← 2-Opt(offspring);
11:          sol_Int ← 1-Interchange(offspring);
12:          offspring ← Best(offspring, sol_Opt, sol_Int);
13:          Insert_If_Better(offspring, cga,aux_pop, x, y);
14:       end for
15:    end for
16:    cga.pop ← aux_pop;
17:    Update_Statistics(cga);
18: end while
19: end_proc Steps_Up;
```

Algorithm 1. Pseudocode of JCell2o1i.

The crossover and mutation operators are applied to the individuals in lines 8 and 9, respectively, with specified probabilities. After that, a local search technique—see below—is applied to every individual (lines 10 to 12). Finally, after applying the above mentioned operators, the offspring is inserted in the new (auxiliary) population—line 13—only if its fitness value is larger than that of the parent located at that position in the current population (elitist replacement).

After each generation (when all the individuals have been considered), the old population is replaced by the new one at once (line 16), and then some statistics are computed (line 17), e.g., maximum, minimum, and average fitness values. The algorithm stops when an a priori predetermined maximum number of generations is reached.

We use the edge recombination operator (ERX [9]), and the mutation consists in applying with equal probability three different mutation methods typically found in routing problems. Our idea here has been to merge them in a new *combined* mutation. These mutation methods are *Insertion* [10], *Swap* [11], and *Inversion* [12]. In all the three operators, the induced changes might occur in an intra or inter-route way.

The local search step consists in applying to the individual 2-Opt [13] and 1-Interchange [14], which are two of the most successful techniques in the past years. This local search returns the best individual among the input one and the output of 2-Opt and 1-Interchange. The fitness value assigned to every individual is computed as follows [2]:

$$f(\vec{S}) = F(\vec{S}) + \lambda \cdot \mathrm{overcap}(\vec{S}) + \mu \cdot \mathrm{overtm}(\vec{S}), \tag{2}$$

$$f_{\mathrm{eval}}(\vec{S}) = f_{\max} - f(\vec{S}). \tag{3}$$

The objective of our algorithm is to maximize $f_{\mathrm{eval}}(\vec{S})$ (Eq. (3)) by minimizing $f(\vec{S})$ (Eq. (2)). The value $f_{\max}$ must be larger or equal with respect to that of the worst feasible solution for the problem. Function $f(\vec{S})$ is calculated by adding the total costs of all the routes ($F(\vec{S})$—see Eq. (1)), and penalizes the fitness value if the capacity of any vehicle and/or the time of any route are exceeded. Functions 'overcap($\vec{S}$)' and 'overtm($\vec{S}$)' return the overhead in capacity and time, respectively, of the solution with respect to the maximum allowed value of each route. The values returned by 'overcap($\vec{S}$)' and 'overtm($\vec{S}$)' are weighted by multiplying them by the values λ and μ, respectively. We here use $\lambda = 1000$ and $\mu = 1000$.

3. Solving CVRP with JCell2o1i

In this section we describe the results of the experiments we have done for testing our algorithm on CVRP. JCell2o1i has been implemented in Java and tested on a 2.8 GHz PC under Linux with a test suite composed by instances belonging to the benchmarks of Van Breedam [6], Golden et al. [7], and Taillard [8]. All the studied instances are publicly available at [15], as well as our new best solutions. We list in Table 1 the parameters used by JCell2o1i in all our tests.

The benchmark of Van Breedam used is a reduced set of the instances from a larger set of 420 ones he proposed in his thesis [6] (there exist more recent works using this benchmark [16,17], but no new best-solutions are reported in them). These instances are characterized by many different features, like the distribution of the customers (single, clusters, cones, ...) and the depot (central, inside, outside) in the space, time windows, pick-up and delivery, or heterogeneous demands.

Table 1
Parameterization used in JCell2o1i

Population size	100 Individuals (10 × 10)
Selection of parents	BT + Current individual
Recombination	ERX, $p_c = 0.65$
Individual mutation	Combined, $p_m = 0.85$
Probability of bit mutation	$p_{bm} = 1.2$/Individual_Length
Replacement	Rep_if_Better
Local search	2-Opt + 1-Interchange, 20 optimization steps

All the instances in this benchmark contain 100 customers, with a total demand of 1000 units. Problems `Bre-1` to `Bre-6` are constrained only by vehicle capacity and have uniform demands, while `Bre-9` to `Bre-11` have heterogeneous demands at the stops. Finally, problems `Bre-7`, `Bre-8` and `Bre-12` to `Bre-15` are not studied in this work because they include pick-up and delivery or time windows constraints (out of scope).

The benchmark of Golden et al. is composed of 20 large scale instances, ranging from 200 to 483 customers arranged in distinct geometric shapes (a six-pointed star, or concentric circles or squares). The instances are called `gold-nXXX-kXX` and `gol-nXXX-kXX` depending on whether or not they have maximal route length restrictions, respectively. The number followed by the `n` means the number of nodes (customers plus depot), and the `k` stands for the number of routes (vehicles used). The depot can be centered or not.

The third selected benchmark is the one proposed by Taillard in 1993. It is composed of 15 nonuniform problems, and the customers are located in the plane, spread in several clusters. The quantities ordered by the customers are exponentially distributed, so the demand of one customer may require the entire capacity of one vehicle.

In Sections 3.1 and 3.2 we analyze in detail the results obtained with JCell2o1i for the instances composing our test suite. In the former, we study the nine instances whose best-known solutions JCell2o1i was able to improve, while in the latter section we analyze the behavior of the algorithm on the rest of the studied instances.

The numerical results of the algorithms are computed after making 100 independent runs for statistical significance. In the benchmark of Van Breedam, the distances between customers have been rounded to the closest integer value in order to obtain consistent values with those reported in [17], but in the case of the other two benchmarks no rounding has been made to show the accuracy of our algorithm regardless of whether or not distances are rounded.

In this section, the metrics for the performance of the algorithm we use (see Table 2) are the value of the best solution found for each instance, the average number of evaluations needed to get that solution, the average elapsed time, the average value of the solutions found in all the independent runs, the deviation (Δ) between our best solution and the known best one so far for the instance (in percentage), and the previously best-known solution for each instance. The deviation between our

Table 2
Computational facts on the improved instances (benchmarks of Van Breedam and Taillard)

Instance	Best solution found			Avg. solution	Δ (%)	Previously best-known
	Best sol.	Avg. evaluations	Avg. time (s)			
Bre-1	**1106.00**	428896.92 ± 365958.66	1068.78 ± 942.23	1113.03 ± 10.80	3.24	1143.0 [17]
Bre-2	**1506.00**	365154.72 ± 196112.12	898.52 ± 512.54	1513.68 ± 11.07	4.68	1580.0 [17]
Bre-4	**1470.00**	210676.00 ± 82554.15	560.42 ± 222.09	1470.00 ± 0.00	0.41	1476.0 [17]
Bre-5	**950.00**	977540.00 ± 554430.07	2045.11 ± 1175.78	955.36 ± 4.16	2.56	975.0 [17]
Bre-6	**969.00**	820180.39 ± 436314.36	1713.25 ± 919.03	970.79 ± 2.46	1.02	979.0 [17]
Bre-9	**1690.00**	1566300.00 ± 0.0	3750.26 ± 0.0	1708.48 ± 10.48	5.59	1790.0 [17]
Bre-10	**1026.00**	1126575.00 ± 428358.44	2794.96 ± 1027.62	1033.34 ± 6.22	1.82	1045.0 [17]
Bre-11	**1128.00**	1742600.00 ± 840749.96	4258.89 ± 1989.96	1153.35 ± 10.81	2.76	1160.0 [17]
tai75b	**1344.62**	75143.36 ± 20848.50	230.87 ± 54.62	1345.10 ± 0.63	0.00	1344.64 [8]

best solution (*sol*) and the best-so-far one (*best*) is computed by (Eq. (4)).

$$\Delta(sol) = \left[\frac{|best - sol|}{best}\right] * 100. \tag{4}$$

3.1. Improved instances

In this section we focus on the nine instances for which JCell2o1i was able to improve the best-known solution so far. Our results are shown in Table 2. Notice that, since all the results included in that table are better than the best-known ones, the displayed Δ values are, actually, improvements with respect to the best-known values so far. The bold values represent new best optima. The deviation (improvement) between our results and the best-known ones ranges from 0.0015%, obtained for instance tai75b, to 5.5865%, computed in the case of instance Bre-9.

As it can be seen in Table 2, the average solution found in all the runs is very close to the best one. Moreover, it is better than the previously best-known solution in all the cases (except tai75b), what is an indication of the high accuracy and stability of our algorithm.

Finally, in terms of the average elapsed computational times, measured in seconds, it can be seen that the longest times correspond to the instances having heterogeneous demands at the stops (instances Bre-9, Bre-10 and Bre-11). For the other tested instances the reader can notice how the elapsed time grows when the vehicle capacity is increased and, hence, when the size of the routes is incremented.

In Fig. 1 we plot the evolution of the best solution found by JCell2o1i during a typical execution when solving instance Bre-2. As it could be expected, there is a fast convergence towards accurate solutions at the beginning of the execution, probably motivated by the local search step, while in the rest of the search the evolution of the best solution is not that fast. The best-so-far

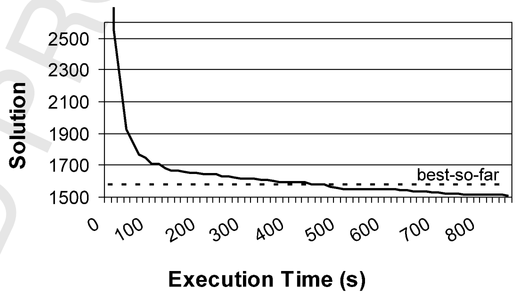

Fig. 1. Trace of an execution for instance Bre-2.

solution is reached at the middle of the execution, approximately, and the new best solution was found after 852 seconds in this run.

3.2. Other instances

We proceed in this section to present the results obtained by JCell2o1i on other instances in the aim of making a fair and wider study. Our results are shown in Table 3. As it can be seen, JCell2o1i finds the best-known solution so far for 5 instances, while in the other cases the deviation between the best-known solution and that reported by the algorithm (Δ) is always very low (under 2.87%). Moreover, the average value of the solutions found is quite close to the best one. Hence, we can conclude that JCell2o1i is also accurate when solving other instances included in the proposed benchmarks.

4. Conclusions and further work

In this article we have improved the best-known solution for nine instances of CVRP. The studied instances have been taken from three different benchmarks, and they are characterized by many distinct features, i.e., uniformly and not uniformly dispersed customers, clustered and not clustered, with a centered or not centered

Table 3
Computational results for the benchmarks of Van Breedam, Taillard et al., and Golden et al.

Instance	Our results		Δ (%)	Best-known	Instance	Our results		Δ (%)	Best-known
	Best sol.	Avg. solution				Best sol.	Avg. solution		
Bre-3	**1751.00**	1751.00 ± 0.00	0.00	**1751.0** [17]	gol-n321-k30	1107.24	1122.32 ± 9.71	2.43	**1081.31** [18]
tai75a	**1618.36**	1619.79 ± 2.18	0.00	**1618.36** [8]	gol-n324-k16	754.95	766.31 ± 6.29	1.74	**742.03** [19]
tai75c	**1291.01**	1294.81 ± 10.02	0.00	**1291.01** [8]	gol-n361-k33	1391.27	1409.42 ± 7.74	1.79	**1366.86** [18]
tai75d	**1365.42**	1365.53 ± 0.88	0.00	**1365.42** [8]	gol-n397-k34	1367.47	1388.93 ± 8.52	1.65	**1345.23** [18]
tai100a	2047.90	2070.39 ± 9.39	0.32	**2041.34** [20]	gol-n400-k18	935.91	950.07 ± 8.57	1.90	**918.45** [18]
tai100b	1940.36	1943.60 ± 4.68	0.02	**1939.90** [18]	gol-n421-k41	1867.30	1890.16 ± 12.28	2.53	**1821.15** [18]
tai100c	1411.66	1419.41 ± 4.05	0.39	**1406.20** [20]	gol-n481-k38	1663.87	1679.55 ± 9.64	2.54	**1622.69** [18]
tai100d	1584.20	1599.57 ± 4.69	0.19	**1581.25** [20]	gol-n484-k19	1126.38	1140.03 ± 7.11	1.73	**1107.19** [18]
tai150a	3056.41	3123.26 ± 51.69	0.04	**3055.23** [8]	gold-n201-k5	**6460.98**	6576.48 ± 94.81	0.00	**6460.98** [21]
tai150b	2732.75	2782.21 ± 28.61	2.87	**2656.47** [20]	gold-n241-k10	5669.82	5789.02 ± 42.09	0.75	**5627.54** [18]
tai150c	2364.08	2409.76 ± 26.12	0.95	**2341.84** [8]	gold-n281-k8	8428.18	8576.61 ± 89.38	0.18	**8412.88** [21]
tai150d	2654.69	2688.54 ± 14.00	0.35	**2645.39** [8]	gold-n321-k10	8488.29	8549.17 ± 47.61	0.48	**8447.92** [21]
tai385	25015.01	25320.96 ± 144.83	2.39	**24431.44** [22]	gold-n361-k9	10227.51	10363.09 ± 66.70	0.31	**10195.56** [21]
gol-n241-k22	714.73	724.96 ± 7.11	0.98	**707.79** [18]	gold-n401-k10	11164.11	11293.47 ± 69.95	1.16	**11036.23** [21]
gol-n253-k27	872.59	884.05 ± 5.98	1.57	**859.11** [18]	gold-n441-k11	11946.05	12035.17 ± 63.45	2.42	**11663.55** [18]
gol-n256-k14	589.20	601.10 ± 5.43	0.99	**583.39** [18]	gold-n481-k12	13846.94	14052.71 ± 140.07	1.63	**13624.52** [21]
gol-n301-k28	1019.19	1037.31 ± 18.59	2.04	**998.73** [18]					

depot, with homogeneous or heterogeneous demands, etc.

We consider that the behavior of JCell2o1i is very satisfactory since, in addition to the number of instances it has been able to improve (up to nine), the solutions found by the algorithm in the other instances are very close to the best existing ones, with variations ranging from 0.00 to 2.87%. Moreover, besides its accuracy, JCell2o1i is quite simple since it is a canonical cGA with three widely used mutations in the literature for this problem, plus two well known local search methods.

As a future work, it may be interesting to test the behavior of the algorithm with other local search methods, i.e., 3-Opt. Other extensions are to test the behavior of JCell2o1i with additional benchmarks of CVRP (ongoing work), to adapt the algorithm for testing it on other variants of VRP, or to add a convergence criterion to the termination condition for avoiding (many) runs without improvement.

Acknowledgements

This work was funded by MCYT and FEDER under contract TIC2002-04498-C05-02 (the TRACER project) http://tracer.lcc.uma.es.

References

[1] G. Dantzig, R. Ramser, The truck dispatching problem, Management Science 6 (1959) 80–91.
[2] E. Alba, B. Dorronsoro, Solving the vehicle routing problem by using cellular genetic algorithms, in: Evolutionary Computation in Combinatorial Optimization (EvoCOP), in: Lecture Notes in Comput. Sci., vol. 3004, Springer-Verlag, Berlin, 2004, pp. 11–20.
[3] B. Manderick, P. Spiessens, Fine-grained parallel genetic algorithm, in: Proc. of the 3rd Int. Conference on Genetic Algorithms (ICGA), Morgan Kaufmann, Los Altos, CA, 1989, pp. 428–433.
[4] D. Whitley, Cellular genetic algorithms, in: Proc. of the 5th Internat. Conference on Genetic Algorithms (ICGA), Morgan Kaufmann, Los Altos, CA, 1993, p. 658.
[5] N. Christofides, A. Mingozzi, P. Toth (Eds.), Combinatorial Optimization, John Wiley, 1979, pp. 315–338 (Ch. The Vehicle Routing Problem).
[6] A.V. Breedam, An analysis of the behavior of heuristics for the vehicle routing problem for a selection of problems with vehicle related, customer-related, and time-related constraints, Ph.D. thesis, University of Antwerp, RUCA, Belgium, 1994.
[7] B. Golden, E. Wasil, J. Kelly, I.-M. Chao, in: Fleet Management and Logistics, Kluwer, Boston, 1998 pp. 33–56 (Ch. The Impact of Metaheuristics on Solving the Vehicle Routing Problem: algorithms, problem sets, and computational results).
[8] E. Taillard, Parallel iterative search methods for vehicle routing problems, Networks 23 (8) (1993) 661–673.
[9] D. Whitley, T. Starkweather, D. Fuquay, Scheduling problems and traveling salesman: The genetic edge recombination operator, in: Proc. of the 3rd Internat. Conference on Genetic Algorithms (ICGA), Morgan Kaufmann, Los Altos, CA, 1989, pp. 133–140.
[10] D. Fogel, An evolutionary approach to the traveling salesman problem, Biological Cybernetics 60 (1988) 139–144.
[11] W. Banzhaf, The "molecular" traveling salesman, Biological Cybernetics 64 (1990) 7–14.
[12] J. Holland, Adaptation in Natural and Artificial Systems, University of Michigan Press, 1975.
[13] G. Croes, A method for solving traveling salesman problems, Operations Research 6 (1958) 791–812.
[14] I. Osman, Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problems, Annals of Operations Research 41 (1993) 421–451.

[15] http://neo.lcc.uma.es/radi-aeb/WebVRP/Problem_Instances/CVRPInstances.html.

[16] A.V. Breedam, Comparing descent heuristics and metaheuristics for the vehicle routing problem, Computers & Operations Research 28 (2001) 289–315.

[17] A.V. Breedam, A parametric analysis of heuristics for the vehicle routing problem with side-constraints, European Journal of Operational Research 137 (2002) 348–370.

[18] D. Mester, O. Bräysy, Active guided evolution strategies for large-scale vehicle routing problems with time windows, Computers & Operations Research 32 (2005) 1593–1614.

[19] F. Li, B. Golden, E. Wasil, Very large-scale vehicle routing: New test problems, algorithms, and results, Computers & Operations Res. 32 (2005) 1165–1179.

[20] L. Gambardella, E. Taillard, G. Agazzi, in: New Ideas in Optimization, McGraw-Hill, 1999, pp. 63–76 (Ch. MACS-VRPTW: A Multiple Ant Colony System for Vehicle Routing Problems with Time Windows).

[21] C. Tarantilis, C. Kiranoudis, Boneroute: an adaptive memory-based method for effective fleet management, Annals of Operations Research 115 (2002) 227–241.

[22] Y. Rochat, E. Taillard, Probabilistic diversification and intensification in local search for vehicle routing, Journal of Heuristics 1 (1995) 147–167.