GVR: a New Genetic Representation for the Vehicle Routing Problem

Francisco B. Pereira[1,2], Jorge Tavares[2], Penousal Machado[1,2], Ernesto Costa[2]

[1]Instituto Superior de Engenharia de Coimbra, Quinta da Nora, 3030 Coimbra, Portugal
[2]Centro de Informática e Sistemas da Universidade de Coimbra, 3030 Coimbra, Portugal
{xico, machado, ernesto}@dei.uc.pt
jast@student.dei.uc.pt

Abstract. In this paper we analyse a new evolutionary approach to the vehicle routing problem. We present Genetic Vehicle Representation (GVR), a two-level representational scheme designed to deal in an effective way with all the information that candidate solutions must encode. Experimental results show that this method is both effective and robust, allowing the discovery of new best solutions for some well-known benchmarks.

1 Introduction

The Vehicle Routing Problem (VRP) is a complex combinatorial optimization problem, which can be seen as a merge of two well-known problems: the Traveling Salesperson (TSP) and the Bin Packing (BPP). It can be described as follows: given a fleet of vehicles with uniform capacity, a common depot, and several costumer demands (represented as a collection of geographical scattered points), find the set of routes with overall minimum route cost which service all the demands. All the itineraries start and end at the depot and they must be designed in such a way that each costumer is served only once and just by one vehicle. VRP is NP-hard, and therefore difficult to solve. Due to its theoretical and practical interest (it has numerous real world applications, given that distribution is a major part of logistics and a substantial cost for many companies), the VRP has received a great amount of attention since its proposal in the 1950's.

Due to the nature of the problem it is not viable to use exact methods for large instances of the VRP (for instances with few nodes, the branch and bound technique [1] is well suited and gives the best possible solution). Therefore, most approaches rely on heuristics that provide approximate solutions. Some specific methods have been developed to this problem (see, e. g., [2], [3]). Another option is to apply standard optimization techniques, such as tabu search [4], simulated annealing [4], [5], constraint programming [6], or ant colony optimization [7].

In the past few years there has also been some applications of evolutionary computation (EC) techniques to the VRP (for a good overview on this topic, consult [8]). Most researchers rely on hybrid approaches that combine the power of an EC algorithm with the use of specific heuristics (see, e.g., [9], [10]) or use simplified versions of the problem. One common simplification is to pre-set the number of

vehicles that is going to be used in the solution [11], [12]. When applied alone, the success of EC techniques has been limited. In the literature we found two reports [13], [14] about the application of non-specific EC methods to wide-ranging versions of this problem (i.e., versions that do not consider any kind of simplification). Both of them were tested in several well-known benchmarks and obtained results that are not very good, when compared to the best solutions found by other approaches.

We consider that the representation adopted for individuals plays a crucial role in the performance of an EC algorithm. In this paper we propose a new representational scheme intended to deal efficiently with the two levels of information that a solution must encode: clustering of the demands (i.e., allocation of all the demands to different vehicles) and specification of the delivery ordering for each one of the routes. This representation also enables an easy adjustment of the number of vehicles required for one possible solution. The search process relies on standard EC techniques. We do not use specific heuristics, nor do we perform any kind of simplification to the problem.

The paper has the following structure: in section 2 we give a formal definition of the VRP. Section 3 comprises a description of the proposed EC model. In section 4 we present and analyze the most important experimental results achieved. Finally, in section 5, we draw some overall conclusions and suggest directions for future work.

2 The Vehicle Routing Problem

The most general version of the VRP is the Capacitated Vehicle Routing Problem (CVRP), which can be formally described in the following way: there is one central depot 0, which uses k independent delivery vehicles, with identical delivery capacity C, to service demands d_i from n customers, $i = 1, \ldots, n$. The vehicles must accomplish the delivery with a minimum total length cost, where the cost c_{ij} is the distance from customer i to customer j, with $i, j \in [1, n]$. The distance between customers is symmetric, i.e., $c_{ij}=c_{ji}$ and also $c_{ii}=0$. A solution for the CVRP would be a partition $\{R_1,\ldots,R_k\}$ of the n demands into k routes, each route R_q satisfying $\sum_{p \in R_q} d_p \leq C$. Associated with each R_q is a permutation of the demands belonging to it, specifying the delivery order of the vehicles. In figure 1 we present an illustration of the problem, viewed as a graph, where the nodes represent the customers.

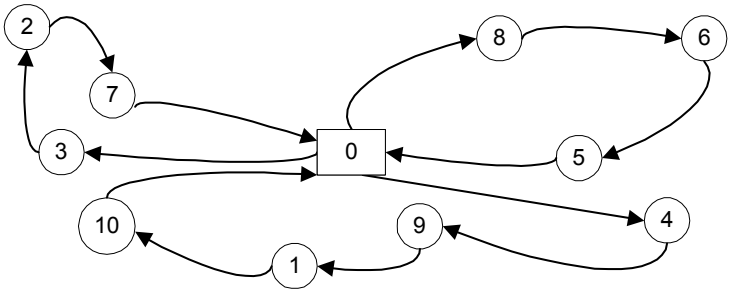

Figure 1: Vehicle Routing Problem

3 Experimental Model

3.1 Genetic Vehicle Representation (GVR)

A candidate solution to an instance of the CVRP must specify the number of vehicles required, the partition of the demands through all these vehicles and also the delivery order for each route. We adopted a representation where the genetic material of an individual contains several routes, each one of them composed by an ordered subset of the costumers. All demands belonging to the problem being solved must be present in one of the routes. As an example, the chromosome from figure 2 represents the solution presented in figure 1.

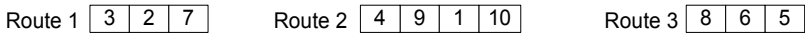

Figure 2: An example of a GVR chromosome.

Some routes in the chromosome may cause a vehicle to exceed its capacity. When this happens, and to guarantee that the interpretation yields a valid solution, we split the route that exceeds capacity in several ones. An example illustrates this adjustment: assume that the original route {a, b, c, d, e, f} causes the vehicle to exceed its capacity at node d. When this situation occurs, the itinerary is divided in two sections: {a, b, c} and {d, e, f}, and a new vehicle is added to the solution. If necessary, further divisions can be made in the second section. Notice that these changes only occur at the interpretation level and, therefore, the information codified in the chromosome is not altered.

3.2 Genetic Operators

The EC algorithm processes the individuals in a straightforward way. Assuming that the population size is N, in each generation N parents are chosen and N descendants are obtained through the application of genetic operators to the elements of the selected set.

We consider two categories of operators: crossover and mutation. They must be able to deal with the two levels of the representation. Thus, they should be capable to change the delivery order within a specific route and to modify the allocation of demands to vehicles. In this last situation, they can, not only switch costumers from one route to another, but also modify the number of vehicles belonging to a solution (adding and removing routes). Another important requirement is that the genetic operators must always generate legal solutions.

The crossover operator used in our approach does not promote a mutual exchange of genetic material between two parents. Instead, when submitted to this kind of operation, one individual receives a fragment of genetic material (more precisely, a sub-route) from another parent and inserts it in one of its own routes. The donor is not modified. The geographical location of the costumers is used to determine the position where the sub-route is inserted. The following algorithm clarifies how crossover is applied to the individuals of the selected set:

```
1.  For each individual I1 from the selected set S Repeat:
    1.1. Randomly select another individual I2 from S
    1.2. From the genetic material of I2 randomly select a sub-route SR:
         {a1, a2, …, an}
    1.3. Find the costumer c, not belonging to SR, that is
         geographically closer to a1
    1.4. Insert SR in the genetic material of I1 in such a way that a1 is
         placed immediately after c
    1.5. From the original genetic material of I1 remove all duplicated
         costumers that also appear on SR, obtaining a descendant D
```

The example from figure 3 helps to illustrate how crossover acts. It assumes that costumer 6 is the one that is geographically closer to costumer 9. This way, sub-route {9, 1, 10} is selected from I_2 and is inserted in one the routes of I_1, immediately after demand 6. The crossover operator makes possible that a fragment of information that is part of one individual can be incorporated into another solution. Given its behaviour, it cannot add new vehicles to the solution. On the other hand, it might remove routes.

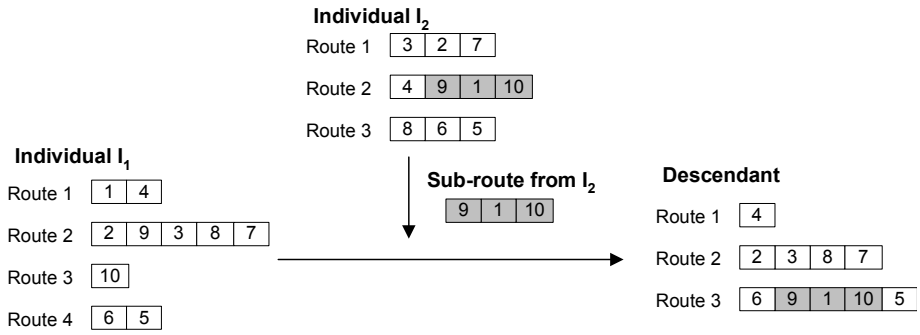

Figure 3: Example of crossover.

Descendants resulting from crossover can be subject to mutation. We consider 4 operators, based on proposals usually applied to order-based representations:

Swap: selects two costumers and swaps them. Selected points can belong to the same or to different routes.

Inversion: selects a sub-route and reverses the visiting order of the costumers belonging to it.

Insertion: selects a costumer and inserts it in another place. The route where it is inserted is selected randomly. It is possible to create a new itinerary with this single costumer. In all experiments reported in this paper, the probability of creating a new route is $1/(2\times V)$, where V represents the number of vehicles of the current solution. This way, the probability of creating a new route is inversely proportional to the number of vehicles already used. In figure 4 we show an example of this operation.

Displacement: selects a sub-route and inserts it in another place. This operator can perform intra or inter displacement (whether the selected fragment is inserted in the same or in another route). Just like in the previous operator, it is also possible to create a new route with the subsequence (the probability of this occurrence is calculated in the same way).

Swap and inversion do not change the number of routes of an individual. On the contrary, insertion and displacement have the ability to remove and to add vehicles to a solution. All genetic operators described have a specific probability of application.

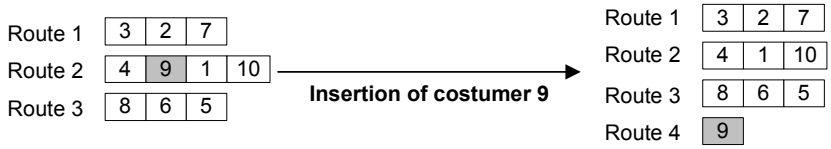

Figure 4: Example of insertion mutation applied over costumer 9.

4 Experimental Results

To evaluate our approach we performed an extensive collection of tests with 12 instances from some well-known benchmarks: Augerat Set A (instances A32k5, A54k7, A60k9, A69k9, A80k10), Augerat Set B (instances B57k7, B63k10, B78k10) and Christofides and Eilon (instances E76k7, E76k8, E76k10, E76k14) [15].

The settings of the EC algorithm are the following: Number of generations: 50000 (except for the instance A32k5, where only 10000 generations were required); Population size: 200; Tournament selection with tourney size: 5; Elitist strategy; Crossover rate: 0.75; Mutation rates: swap: 0.05; inversion: {0.1, 0.15}; insertion: 0.05; displacement: {0.15, 0.2}. For every set of parameters we performed 30 runs with the same initial conditions and with different random seeds. All initial populations were randomly generated according to the following algorithm:

```
1. Build a set D with a random permutation of all the demands
2. While there are elements in D Repeat:
   2.1. Create one new route with K demands. K is a random value
        between 1 and the current number of elements from D
   2.2. Remove first K elements from D and assign them to the new route
```

Even though values for different parameters were set heuristically, we performed some additional tests and verified that, within a moderate range, there was no significant difference in the outcomes.

In table 1 we present, for all instances, the best solution found with each one of the 4 settings selected for this paper. Column “Nr. Points” identifies how many costumers each instance has and column “Previous Best” summarizes the cost of the best solutions that were known when we started our research. A brief perusal of the results reveals that our EC approach was able to consistently find good solutions to the different instances used in the experiments. In six of them (A60k9, A69k9, B57k7, B78k10, E76k10 and E76k14), it discovered new best solutions with lower cost than those ones previously known. It is interesting to notice that in instance B57k7 the new solution found requires 8 vehicles, whereas the previous best required just 7 vehicles but had a higher overall cost (1153 vs. 1140). Since in the CVRP, the cost is the single objective to minimize, the solution we found is clearly better.

In table 2 we show, for the same settings, the average of best solutions found in each one of the 30 runs. Columns labelled “Dist.” present the distance (in percentage) between these averages and the best-known solutions for the instances. Information

from this table reveals that the representation used is very reliable. Considering the results as a whole, the distance that we just mentioned never exceeds 3.5% (in many situations, this value is considerably smaller). A detailed consult to both tables also shows that a small variation in the mutation rates of displacement and inversion does not produce major divergences in the results achieved. Other settings not presented here due to lack of space follow the same trend, showing that this method is robust: it was both able to find new best solutions for some of the instances used in the tests and to guarantee, with high likelihood, that good solutions are found during the search process.

Instances	Nr. Points	Previous Best	Inversion = 0.1		Inversion = 0.15	
			Disp = 0.15	Disp = 0.2	Disp = 0.15	Disp = 0.2
A32k5	32	784	784	784	784	784
A54k7	54	1167	1167	1167	1167	1167
A60k9	60	1358	1358	**1354**	1358	1358
A69k9	69	1167	**1159**	1165	1165	1165
A80k10	80	1764	1764	1777	1781	1774
B57k7	57	1153	**1140**	**1140**	**1140**	**1140**
B63k10	63	1496	1507	1496	1516	1497
B78k10	78	1266	1224	1223	1224	**1221**
E76k7	76	682	683	687	691	683
E76k8	76	735	737	738	738	740
E76k10	76	832	837	841	**830**	837
E76k14	76	1032	1028	**1022**	1031	1030

Table 1: Best solutions found. Bold entries highlight new best solutions.

Instances	Inversion = 0.1				Inversion = 0.15			
	Disp = 0.15		Disp = 0.2		Disp = 0.15		Disp = 0.2	
	Avg.	Dist.	Avg.	Dist.	Avg.	Dist.	Avg.	Dist.
A32k5	793.4	1.2	790.9	0.9	796.5	1.6	786.3	0.3
A54k7	1178.6	1.0	1186.2	1.6	1182.4	1.3	1188.9	1.9
A60k9	1377.9	1.8	1377.9	1.8	1387.9	2.5	1372.4	1.4
A69k9	1180.6	1.9	1182.0	2.0	1180.0	1.8	1179.9	1.8
A80k10	1811.2	2.7	1813.8	2.9	1819.3	3.2	1811.0	2.7
B57k7	1141.1	0.1	1141.2	0.1	1141.7	0.1	1140.7	0.1
B63k10	1546.6	3.4	1544.0	3.2	1547.0	3.4	1545.7	3.3
B78k10	1255.3	2.8	1254.6	2.7	1249.5	2.3	1252.3	2.6
E76k7	705.9	3.5	704.8	3.3	703.4	3.1	705.0	3.4
E76k8	755.9	2.8	755.3	2.8	756.2	2.9	755.4	2.8
E76k10	851.7	2.5	856.6	3.2	852.6	2.7	854.0	2.9
E76k14	1042.9	2.0	1043.5	2.1	1045.2	2.3	1043.0	2.1

Table 2: Average of the best solutions found in each one of the 30 runs.

Another important feature of the algorithm is that it shows a good ability to escape from premature convergence to regions containing local optima. An example helps to illustrate this situation: we selected the instance with a higher number of costumers (A80k10) and extended the search process to 100000 generations. Table 3 shows the effect of the duplication of the simulation time. With all setting, the average of the best solutions found in the 30 runs shows a slight decrease and, most important, it was possible to find a new best solution with cost 1763. This example confirms that search

is not stagnated. Even though it is possible to discover good solutions in early stages of the search process, if the algorithm is given enough time to carry on its exploration, it might continue to improve the best individuals found.

Nr. of generations	Inversion = 0.1				Inversion = 0.15			
	Disp = 0.15		Disp = 0.2		Disp = 0.15		Disp = 0.2	
	Best	Avg.	Best	Avg.	Best	Avg.	Best	Avg.
50000	1764	1811.2	1777	1813.8	1781	1819.3	1774	1811.0
100000	1764	1800.8	1777	1808.6	1775	1813.4	**1763**	1805.8

Table 3: Effect of the duplication of the simulation time with instance A80k10. The bold entry highlights a new best solution.

In this paper we will not perform a detailed analysis about the relative importance of each one of the 5 genetic operators used. This study will be carried out in a future publication. We want nevertheless to illustrate how crossover is important to discover good solutions. To achieve this goal, we performed an additional set of experiments that did not use this operator. Displacement mutation was selected to replace crossover. In table 4 we present results of experiments performed with the following mutation rates: insertion and swap: 0.05; inversion: 0.15; displacement: {0.2, 0.5, 0.75}. All other settings remain unchanged. To help comparison, column "With Cx." shows the results achieved in the experiment performed with the following settings: crossover rate: 0.75; insertion and swap: 0.05; inversion: 0.15; displacement: 0.2. Even though this is just a preliminary study, a brief inspection of table 4 reveals that the transfer of genetic material between parents seems to be crucial to the success of the proposed approach. Solutions found by experiments that used crossover are consistently better than all other situations that just relied on mutation. If we maintain our focus on the results presented on table 4 and consider each one of the 12 instances separately, in most cases the average of the best solutions found by the experiment that used crossover is significantly lower (significance level: 0.05) than those ones achieved by experiments that just relied in mutation. The only exceptions occur in the 3 experiments performed with instance B57k7 and in one of the experiments done with instance A57k7 (the one that used displacement rate = 0.5).

Instances	Disp = 0.2		Disp = 0.5		Disp = 0.75		With Cx.	
	Best	Avg.	Best	Avg.	Best	Avg.	Best	Avg.
A32k5	784	816.3	784	809.1	784	813.5	784	786.3
A54k7	1176	1219.3	1167	1199.4	1172	1209.4	1167	1188.9
A60k9	1363	1422.5	1358	1414.2	1377	1415.7	1358	1372.4
A69k9	1175	1213.7	1165	1199.6	1172	1209.7	1165	1179.9
A80k10	1801	1783.5	1813	1868.5	1851	1897.0	1174	1811.0
B57k7	1140	1142.7	1140	1141.2	1140	1142.2	1140	1140.7
B63k10	1507	1569.9	1504	1558.4	1548	1571.1	1497	1545.7
B78k10	1253	1289.6	1266	1289.4	1243	1281.4	1221	1252.3
E76k7	692	718.4	692	713.3	692	722.5	683	705.0
E76k8	747	778.3	748	768.7	738	767.4	740	755.4
E76k10	861	881.1	846	866.1	846	869.0	837	854.0
E76k14	1040	1062.9	1044	1064.3	1040	1064.4	1030	1043.0

Table 4: Summary of the influence of crossover on the search process.

5 Conclusions and Further Work

In this paper we presented a new generic evolutionary approach to the VRP. The two-level representational scheme we proposed proved to be extremely effective to this problem, since we were able to find new best solutions for several instances belonging to well-known benchmarks. Moreover, results show that this method is both robust and scalable.

Results presented here can be considered as preliminary. As future work we intend to perform a detailed study on the importance of the genetic operators presented in this paper and also extend our approach to other variants of the problem, such as the VRP with time windows.

References

1. Desrosiers, J., Madsen, O., Solomon, M. and Soumis, F. (1999). 2-Path Cuts for the Vehicle Routing Problem with Time Windows, *Transportation Science*, Vol. 33, No. 1, pp. 101-116.
2. Thangiah, S., Potvin, J. and Sun, T. (1996). Heuristic Approaches to Vehicle Routing with Backhauls and Time Windows, *Int. Journal of Computers and Operations Research*, pp. 1043-1057.
3. Prosser, P. and Shaw, P. (1997). Study of Greedy Search with Multiple Improvement Heuristics for Vehicle Routing Problems, *Technical Report RR/96/201*, Department of Computer Science, University of Strathclyde, Glasgow.
4. Tan, K. C., Lee, L. H., Zhu, Q. L. and Ou K. (2000). Heuristic Methods for Vehicle Routing Problem with Time Windows, *Artificial Intelligent in Engineering*, pp. 281-295.
5. Bent, R. and Hentenryck, P. V. (2001). A Two-Stage Hybrid Local Search for the Vehicle Routing Problem with Time Windows, *Technical Report, CS-01-06*, Brown University.
6. Shaw, P. (1998). Using Constraint Programming and Local Search Methods to Solve Vehicle Routing Problems, *Proceedings of the Fourth International Conference on Principles and Practice of Constraint Programming (CP '98)*, M. Maher and J. Puget (eds.), pp. 417-431.
7. Gambardella, L. M., Taillard, E. and Agazzi, G. (1999) MACS-VRPTW: A Multiple Ant Colony System for Vehicle Routing Problems with Time Windows, In D. Corne, M. Dorigo and F. Glover (eds.), *New Ideas in Optimization*. McGraw-Hill, London, UK, pp. 63-76.
8. Bräysy, O. (2001). Genetic Algorithm for the Vehicle Routing Problem with Time Windows, Arpakannus 1/2001, *Special Issue on Bioinformatics and Genetic Algorithms*, pp.33-38.
9. Thangiah, S. R. (1995). Vehicle Routing with Time Windows Using Genetic Algorithms, *Application Handbook of Genetic Algorithms: New Frontiers*, Volume II. Chambers, L.(ed), pp. 253-277, CRC Press.
10. Potvin, J. , Dubé, D. and Robillard, C. (1996). Hybrid Approach to Vehicle Routing Using Neural Networks and Genetic Algorithms, *Applied Intelligence*, Vol. 6, No. 3, pp. 241-252.
11. Zhu, K. (2000). A New Genetic Algorithm for VRPTW, *International Conference on Artificial Intelligence*, Las Vegas, USA.
12. Louis, S. J., Yin, X. and Yuan, Z. Y. (1999). Multiple Vehicle Routing With Time Windows Using Genetic Algorithms, *Proceedings of the Congress of Evolutionary Computation (CEC-99)*, pp. 1804-1808.
13. Duncan, T. (1995). Experiments in the Use of Neighbourhood Search Techniques for Vehicle Routing. *Report AIAI-TR-176*, University of Edinburgh.
14. Machado, P., Tavares, J., Pereira, F. B. and Costa, E. (2002). Vehicle Routing Problem: Doing it the Evolutionary Way, *To appear at GECCO-2002 Proceedings*.
15. Available at: *http://www.branchandcut.org/VRP/data/*