Tuning a Parametric Clarke-Wright Heuristic via a Genetic Algorithm

Maria Battarra[*], Bruce Golden[+], Daniele Vigo[*]

[*] Università di Bologna, Dipartimento di Elettronica Informatica e Sistemistica

Via Venezia, 52, - 47023 Cesena, Italy, {mbattarra,dvigo}@deis.unibo.it

[+]Robert H. Smith School of Business, University of Maryland, College Park, MD 20742, U.S.A., BGolden@rhsmith.umd.edu

Technical Report DEIS OR.INGCE 2006/1R
July 2006

Abstract

Almost all heuristic optimization procedures require the presence of a well-tuned set of parameters. The tuning of these parameters is usually a critical issue and may entail intensive computational requirements. We propose a fast and effective approach composed of two distinct stages. In the first stage, a genetic algorithm is applied to a small subset of representative problems to determine a few robust parameter sets. In the second stage, these sets of parameters are the starting points for a fast local search procedure, able to more deeply investigate the space of parameter sets for each problem to be solved. This method is tested on a parametric version of the Clarke and Wright algorithm and the results are compared with an enumerative parameter setting approach previously proposed in the literature. The results of our computational testing show that our new parameter-setting procedure produces results of the same quality as the enumerative approach, but requires much shorter computational time.

Keywords*:* Vehicle Routing, Heuristics, Genetic Algorithms.

1. Introduction

Almost all heuristic optimization procedures require the setting of some parameters, such as the weights used in evaluation formulae and the number of iterations to be performed. Determining appropriate values for such parameters, i.e., that allow good performance over all the relevant instances to be solved, is clearly a critical issue in the design and testing of algorithms that may require intensive computational requirements. Therefore, the research on fast and effective approaches for parameter tuning is of considerable importance both from a theoretical and a practical point of view.

The use of genetic algorithms to solve parameter setting problems was first proposed by (Golden *et al,* 1998), who used a two-stage procedure to define the parameter values of a Lagrangian Relaxation based heuristic for the Vehicle Routing Problem (VRP). Given a small set of representative problem instances, called the *analysis set* in the following, in the first stage of the procedure, a genetic algorithm is used to determine, for each separate instance, a good parameter vector. Then, in the second stage, another genetic algorithm determines a set of weights to combine the different parameter vectors into one overall vector. The computational testing, over a larger set of benchmark instances from the literature, showed that this approach produced more robust parameter settings than traditional experimental design methods. The same technique was applied by (Pepper *et al*, 2002) to the parameter setting phase of an annealing-based heuristic for the Travelling Salesman Problem (TSP).

More recently, (Chandran *et al,* 2003) introduced a simpler single-stage procedure in which a genetic algorithm produces the parameters by considering the analysis set as a whole. This approach was applied with very good results by Chandran et al. to a variant of the so-called demon algorithm for the TSP.

In this paper, we further investigate this approach by defining a single stage procedure based on a genetic algorithm for a parametric version, proposed by (Altınel and Öncan, 2005), of the well-known (Clarke and Wright, 1964) heuristic for the VRP. In their approach, Altınel and Öncan run an enhancement of the Clarke and Wright heuristic with several thousands of different parameter vectors and obtained a considerable improvement in terms of overall solution quality with respect to the original Clarke and Wright heuristic. Our genetic-based parameter setting procedure experimentally proved capable of obtaining results of comparable quality with much shorter computing times.

The paper is organized as follows. In Section 2, the VRP is defined and the Altınel and Öncan parametric heuristic is briefly described. Our genetic algorithm for the determination of a good set of parameters is presented in Section 3. The results of an extensive computational testing of the proposed approach are discussed in Section 4.

2. The Vehicle Routing Problem and the Altınel and Öncan parametric heuristic

The Vehicle Routing Problem (VRP) is an important and difficult combinatorial optimization problem (Toth and Vigo, 2001) which requires the determination of an optimal set of routes used by a fleet of vehicles to serve a set of customers, taking into account various operational constraints. The most studied variant of this problem is the so-called Capacitated VRP (CVRP) in which all vehicles are identical and only the vehicle capacity constraints are considered. More precisely, let $V = \{0, \ldots, n\}$ be the set of nodes of a complete undirected graph $G = (V, E)$. Node 0 represents the depot, where the vehicles, each with capacity D, are stationed, whereas the remaining nodes represent the n customers, each associated with a nonnegative demand d_i, $i = 1, \ldots, n$. In addition, for each arc $(i, j) \in E$, let c_{ij} denote the cost of traversing it. The CVRP calls for the determination of a set of routes with minimum total cost such that each customer is visited once by a single vehicle which starts and ends its route at the depot, and whose total demand does not exceed the vehicle capacity.

An early and well-known heuristic proposed for the CVRP is the Clarke and Wright algorithm (Clarke and Wright, 1964). This famous heuristic uses the concept of *savings* to rank merging operations between routes. The savings is a measure of the cost reduction obtained by combining two small routes into one larger route. Given two customers i and j, the associated saving is defined as follows:

(1) $\boldsymbol{s_{ij} = c_{i0} + c_{0j} - c_{ij}}$.

The algorithm starts with a solution in which each customer is served alone on a route. Next, for all customer pairs, the saving is computed and the savings list is sorted from largest to smallest. At each iteration of the algorithm, the next savings is considered, and if the two associated customers which belong to two different routes can be feasibly merged into a new route, then the routes are merged. The Clarke and Wright algorithm is very fast and relatively easy to implement, however, given its greedy, nature the solutions obtained are often of insufficient quality with respect to more sophisticated approaches. In particular, it should be noted that the Clarke and Wright algorithm does not allow control over the final number of routes obtained, since the possibility of route mergings may drastically decrease after the first few iterations in tightly constrained problems. Several attempts have been made in the literature to introduce modified parameter-dependent savings formulae that yield better overall results. (Gaskell, 1967) and (Yellow, 1970) introduced the parameter λ (called the *route shape* parameter) which controls the relative importance of the direct arc between the two customers in the savings computation. (Paessens, 1988) included in the saving expression a new term, together with a weight parameter μ, which takes into account the “asymmetry” in terms of distance from the depot to each of the two “merged” customers.

The resulting savings formula is:

(2) $s_{ij} = c_{i0} + c_{0j} - \lambda c_{ij} + \mu \left| c_{0i} - c_{j0} \right|$.

Altınel and Öncan further modified the parametric savings expression in an attempt to consider the twofold nature of the CVRP. In particular, the CVRP combines the structure of a multiple TSP (*m*-TSP) and that of the Bin Packing Problem (BPP). The previously proposed savings formula did not consider the benefit of fully utilizing a vehicle's capacity, hence Altınel and Öncan introduced a third weighted term in the savings formula to focus on customer demands, according to a "*larger combined route is better*" idea:

$$(3) \qquad s_{ij} = c_{i0} + c_{0j} - \lambda c_{ij} + \mu \left| c_{0i} - c_{j0} \right| + \nu (d_i + d_j) / \overline{d} \, .$$

where $\overline{d}$ is the average customer demand.

The new expression requires the tuning of the three independent parameters (λ, μ,ν). To this end, Altınel and Öncan used a simple enumerative approach. They executed the algorithm with parameter values varying in the ranges [0.1, 2] for parameter λ and [0, 2] for parameters μ and ν, using a step size equal to 0.1. In this way, 8820 different parameter vectors are obtained. The resulting enumerative algorithm proved better than the original Clarke and Wright heuristic which corresponds to a parameter vector of (1,0,0), but clearly requires a much larger computing time which is overly burdensome when the number of customers is large.

3. The parameter setting procedure

In this section, we describe the single-phase parameter setting procedure that we propose to determine good values of the parameters to be used with the (Altınel and Öncan, 2005) savings formula. To this end, we use a genetic algorithm that explores the parameter space so as to determine a small subset of vectors *Q* which are able to produce the best average results when applied to the instances of the analysis set. Then, for each instance, our overall algorithm, called BGV, applies the Altınel and Öncan parametric savings algorithm using only the parameter vectors belonging to set *Q* and those obtained with a limited local search step. As already done in previous works from the literature such as (Golden, 1998), (Pepper, 2002) and (Chandran, 2003), a genetic algorithm is used for the tuning process since this type of algorithm is particularly well-suited for the solution of non-constrained optimization problems, such as the tuning problem we are considering.

Genetic algorithms belong to the class of metaheuristics and are adaptive procedures inspired by principles of natural selection and survival of the fittest: see (Holland, 1992) and (Michalewicz, 1996) for a complete introduction. The goal of a genetic algorithm is the optimization of a *fitness function*, which expresses the performance of a specific solution. A genetic algorithm starts with an initial population of solutions (usually generated at random). During a number of iterations or *generations*, the solution space is explored. As with evolution and natural selection, the individuals with the most promising characteristics and the best fitness values are more likely to engage in reproduction and are more likely to survive. Each iteration of a genetic algorithm consists of the following steps:

1) Crossover. The most promising individuals in the population, in terms of fitness value, are chosen as parents for reproduction. Offspring are produced and they reflect the genetic heritage of their parents.

2) Mutation. The genetic characteristics of every offspring are modified following probabilistic rules.

3) Selection. Every offspring, after mutation, is evaluated in terms of its fitness. If this value is high, the offspring is introduced in the population (and another less fit individual is typically expelled), otherwise the offspring is aborted.

Steps 1-3 are repeated until a given stopping criterion is satisfied.

As previously mentioned, a single stage parameter-setting procedure is applied to a subset of problem instances, called the *analysis set*. The instances chosen in the analysis set should be representative of the CVRP instances to be solved but their number should be kept small so as to require a moderate computing time to evaluate the fitness function for each parameter set at a given generation of the genetic algorithm. As illustrated later, in our computational testing we used an analysis set made up of 5 instances chosen from different testing sets.

In our parameter-setting genetic algorithm, the individuals of the population correspond to different vectors of the three parameters (λ, μ, ν) in the (Altınel and Öncan, 2005) savings formula (3). The initial population contains 25 individuals obtained by a random choice of each parameter within the ranges used by Altınel and Öncan, i.e., [0.1, 2] for parameter λ and [0, 2] for parameters μ and ν. The fitness function considered by our genetic algorithm is that proposed by (Chandran *et al,* 2003). For a given parameter vector *w*, this function measures the average, over all instances $i \in P$ of the analysis set, of the squared relative excess deviation, of the solutions $D(w,i)$ obtained with parameter vector *w*, with respect to the best known solution for instance *i* denoted by $B(i)$. Note that the ratio $D(w,i)/B(i)$ is not smaller than 1, therefore, 1 is subtracted from the ratio so as to give the excess deviation. The overall value of the fitness function to be minimized is, thus, given by the following expression:

$$(4) \qquad \boldsymbol{F}(\boldsymbol{w}) = 100 \cdot \sqrt{\frac{1}{|\boldsymbol{P}|} \cdot \sum_{\boldsymbol{i}=1}^{|\boldsymbol{P}|} ((\boldsymbol{D}(\boldsymbol{w},\boldsymbol{i}) / \boldsymbol{B}(\boldsymbol{i})) - 1)^2} \ .$$

The genetic algorithm iteratively evolves the individuals of the population to obtain the individuals of the next generation by means of the following operators:

1) Crossover. The best individual in the population, called the *queen mother*, is identified. An offspring is generated as a product of the queen mother and each other individual in the population. These other individuals are the fathers. In a given

offspring, the value of each parameter is randomly generated from a uniform distribution in the range defined by the corresponding parameter values of the parents.

2) Mutation. For each offspring, the value of each parameter is mutated with probability *r*, by randomly generating a new value according to a uniform distribution in the original range.

3) Selection. The best 25 individuals, in terms of fitness value, of the intermediate population made up of the current set of 25 parents plus the 24 offspring generated by the two previous operators, are selected as the current population for the next iteration of the algorithm.

The iterative process is stopped as soon as the average fitness value of the individuals in the population has not decreased in the last *h* generations.

To determine the set *Q* of parameter vectors to be used in our overall algorithm, we proceed as follows. During the genetic algorithm, we store all the different parameter vectors that were generated. At the end of the execution, we select the vector with the smallest fitness value. We select at most $|Q|$-1 additional vectors whose fitness values are not greater than 20% higher than the best one and that maximize the "distance" from the best vector. As a distance between two vectors, we used the sum of the absolute differences of the corresponding parameter values.

In our heuristic BGV, we run the Clarke and Wright algorithm with the (Altınel and Öncan, 2005) savings formula with a small number of parameter vectors, including those in set Q and some obtained by performing a limited local search around them. More precisely, the local search considers a small neighbourhood including the 14 points corresponding to the 8 vertices and the 6 centres of the faces of a cube whose edge length is 0.2 and whose centre is at the point whose coordinates are the parameter values of the current vector. The parametric Clarke and Wright is executed for each vector in the neighbourhood and the best one not yet visited is selected as new current vector. For each vector of set Q, the local search is iterated three times obtaining 41 different vectors, namely the starting one, plus the 14 vectors of the first neighbourhood and 13 vectors of each subsequent neighbourhood. Since $|Q|=5$, our overall algorithm requires only 205 executions of the parametric Clarke and Wright algorithm. This value is more than 40 times smaller than the number of executions required by the enumerative procedure of (Altınel and Öncan, 2005).

4. Computational results

In this section, we discuss the results obtained during an extensive computational analysis of the BGV heuristic described in this paper. The main aim of this analysis is to verify whether the single-phase parameter setting approach that generates the small subset of parameter vectors used by BGV is able to find vectors that produce, with considerably smaller computing time, solutions having comparable quality to those that can be obtained by the enumerative parameter-setting approach proposed by (Altınel and Öncan, 2005).

In our experiments, we used the same CVRP test instances considered by (Altınel and Öncan, 2005) as a benchmark set, namely the seven instances of (Christofides *et al*, 1969), eight instances of (Christofides and Eilon, 1969), and 72 instances of the three classes proposed by (Augerat *et al*, 1995). All these instances may be downloaded at the site http://www.branchandcut.org. As an analysis set, we selected (from the benchmark set) five instances differing in terms of number of customers and spatial location of the depot. The analysis set is made up of the instances indicated in Table 1.

Name	*n*+1	*D*	Test set	Depot Location	Best Known Solution
P-n16-k8	16	35	(Augerat *et al,* 1995), set P	bottom-left	450
A-n53-k7	53	100	(Augerat *et al,* 1995), set A	centre-left	1010
B-n78-k10	78	100	(Augerat *et al,* 1995), set B	bottom-centre	1221
E-n101-k14	101	112	(Christofides and Elion, 1969)	centre	1067
CMT199	199	200	(Christofides *et al*, 1979)	centre	1291.45

Table 1 The instances of the analysis set

The genetic parameter setting algorithm was coded in Matlab and run on a PC with AMD Athlon TM XP Processor 3000+, 797 MHz and 256 MB RAM. Using this algorithm, with $|Q| = 5$ we determined, within about 12 minutes of computing time, the five parameter vectors reported in Table 2.

Vector	Fitness value	Distance from vector 1	λ	μ	ν
1	5.0564	-	1.5578	0.6920	0.8190
2	5.3053	1.4457	0.8830	0.6948	1.5871
3	5.3704	1.3165	0.7335	0.6657	1.2849
4	5.9371	1.1883	1.4891	0.6404	1.8870
5	5.4035	0.9719	1.6442	0.7251	1.6714

Table 2 The |Q| parameter vectors, produced by the genetic algorithm

The results of the algorithm BGV when applied to the entire set of 87 benchmark instances are summarized in Table 3 and compared with those of the enumerative parameter setting algorithm of (Altınel and Öncan, 2005). More precisely, for each of the five classes of instances in the benchmark set and for each algorithm, the table reports the average computing time in seconds, and the average percentage improvement of the solutions obtained with the algorithm over those obtained from the original Clarke and Wright algorithm. As to the BGV approach, we report both the results obtained by using just the five parameter vectors of Table 2 and with the limited local search (LS) step. To correctly compare the computing times, we use our implementation of the (Altınel and Öncan, 2005) algorithm and all experiments are run on a PC with AMD Athlon TM XP Processor 3000+, 797 MHz, 256 MB RAM. The average percentage improvements of our implementation are nearly equal to those reported in the Altınel and Öncan paper. Finally, the last row of the table gives the average results over all 87 instances of the benchmark set.

Test set	# of instances	Altınel and Öncan		BGV		BGV with Local Search	
		Average Time (sec)	% solution improvement over CW	Average Time (sec)	% solution improvement over CW	Average Time (sec)	% solution improvement over CW
(Augerat *et al,* 1995), set A	27	35.76	2.44	0.02	1.13	0.97	1.84
(Augerat *et al,* 1995), set B	23	39.27	2.10	0.01	0.88	1.06	1.80
(Augerat *et al,* 1995), set P	22	33.20	4.35	0.02	2.52	0.88	1.96
(Christofides and Elion, 1969)	8	50.12	3.05	0.03	1.84	1.25	1.96
(Christofides *et al*, 1979)	7	179.26	3.06	0.10	1.07	4.37	1.96
Total	87	48.91	2.94	0.02	1.47	1.27	2.22

Table 3 Comparison of the parameter setting method with and without Local Search with respect to Altınel and Öncan approach.

By examining Table 3, it can be observed that the BGV algorithm, without the local search step, is more than three orders of magnitude faster than that of Altınel and Öncan and is still able to obtain about half of the improvement relative to the original Clarke and Wright approach. Indeed, it improves on the CW by about 1.5% while Altınel and Öncan improves by less than 3%. By introducing the limited local search step, the computing times are still about 40 times smaller and the improvement is practically the same as that of Altınel and Öncan. In both cases, we did not include the computing time required by the genetic parameter settings procedure which is run once for all instances. On the other hand, even if we include the parameter setting time, equal to 720 seconds in our Matlab implementation, the total time required by the BGV approach for the whole set of 87 instances is still more than five time faster than Altınel and Öncan, which requires more than 4255 seconds in total.

We should observe that the computing time of the parameter tuning procedure may be considerably shortened by using an efficient implementation in the C language rather than the Matlab code that we used. Moreover, given the considerable variety of the instances that were used in the computational testing, it is likely that the set of parameters determined during the parameter tuning can be used with other instances with comparable results, not requiring specific tuning steps.

Ackowledgments

Work partially supported by Ministero dell’Università e della Ricerca, Italy.

References

Altınel I K, Öncan T (2003*). A new enhancement of the Clarke and Wright savings heuristic for the capacitated vehicle routing problem. FBE-IE-02/2003-02*. Institute for Graduate Studies in Science and Engineering, Bogaziçi University, Bebek, Istanbul, Turkiye.

Altınel I K, Öncan T (2005). A new enhancement of the Clarke and Wright savings heuristic for the capacitated vehicle routing problem. *J Opl Res Soc* **56**: 954-961.

Augerat P, Belenguer J M, Benavent E, Corberan A, Naddef D, Rinaldi G (1995). *Computational results with a branch and cut code for the capacitated vehicle routing problem.* Technical report RR 949-M, University Joseph Fourier, Grenoble, France.

Chandran B, Golden B, Wasil E (2003). A Computational Study of Three Demon Algorithm Variants for Solving the Travelling Salesman Problem. In: Barghava HK and Ye N (eds). *Computational Modelling and Problem Solving in the Networked World*: *Interfaces in Computer Science and Operations Research.* Operations Research/Computer Science Interfaces Series, Kluwer Academic Publisher: Boston, MA, pp 155-175.

Christofides N, Eilon S (1969). An algorithm for the vehicle routing dispatching problem. *Opl Res Quart* **20:** 309-318.

Christofides N, Mingozzi A, Toth P(1979). The vehicle routing problem. In: Christofides N, Mingozzi A, Toth P, Sandi C (eds). In: *Combinatorial Optimization.* Wiley, Chichester, pp 315-338.

Clarke G, Wright J (1964). Scheduling of vehicles from a central depot to a number of delivery points. *Opns Res* **12 (4)**: 568-581.

Coy S (1998). *Fine-tuned Learning: A New Approach to Improving the Performance of Local Search Heuristics*. Ph.D.Dissertation, Univ.of Md.,College Park.

Gaskell T J (1967). Bases for vehicle fleet scheduling. *Opl Res Quart* **18**: 281-295.

Golden B, Pepper J, Vossen T (1998). Using genetic algorithms for setting parameter values in heuristic search. In: *Intelligent Engineering System through Artificial Neural Networks*, ASME Press, New York, **8**: 239-245.

Holland H (1992). *Adaptation in Natural and Artificial Systems*. MIT Press: Ann Arbor, MI.

Michalewicz Z (1996). *Genetic Algorithms + Data Structures = Evolution Programs, Third, revised and Extended Edition*. Springer-Verlag Berlin Heidelberg New York Publishing: Berlin.

Paessens H (1988). The savings algorithm for the vehicle routing problem. *Eur J Opl Res* **34**: 336-344.

Pepper J, Golden B, Wasil E (2002). Solving the travel salesman problem with annealing-based heuristics: a Computational Study. IEEE *Transactions on Systems, Man and Cybernetics A* **32(1)**: 72-77.

Toth P, Vigo D (2001). *The vehicle routing problem*. SIAM Monographs on Discrete Mathematics and Applications. SIAM Publishing: Philadelphia, PA.

Yellow P (1970). A computational modification to the savings method of vehicle scheduling. *Ops Res Quart* **21**: 281-283.