BRKGA algorithm for the Capacitated Arc Routing Problem

C. Martinez [a,1,2], I. Loiseau[b,2], M.G.C. Resende[c] and S. Rodriguez[a,1]

[a] *Departamento de Informática, Facultad de Ciencias Exactas Universidad Nacional de Salta, Argentina*

[b] *Departamento de Computación, Facultad de Ciencias Exactas y Naturales, Universidad de Buenos Aires, Argentina*

[c] *AT&T Labs Research, Florham Park - NJ, USA*

Abstract

We propose a new algorithm for the Capacitated Arc Routing Problem (CARP). Our motivation to deal with this problem is related to its application in several real world scenarios such as street sweeping, urban waste collection and electric meter reading just to mention a few.
Based on BRKGA metaheuristic, our algorithm introduces a new random key encoding for CARP, mutation to random keys strings, a restart phase to avoid stagnation and local search .
The algorithm was tested with several well-known instances from the literature. The results obtained were competitive in terms of objective function value and required computational time.

Keywords: BRKGA, CARP, metaheuristics, vehicle routing

1 Introduction

The Capacitated Arc Routing Problem (CARP) consists of designing a minimum cost set of vehicle routes to service a predetermined set of required streets or roads, by means of an homogeneous fleet of vehicles. Each route starts and

[1] Partially supported by CIUNSA 1788-4.
[2] Partially supported by PICT 2006-1600 and UBACyT X143
[3] Email: cmartinez@unsa.edu.ar
[4] Email: irene@dc.uba.ar
[5] Email: mgcr@research.att.com
[6] Email: sryan@unsa.edu.ar

This paper is electronically published in
Electronic Notes in Theoretical Computer Science
URL: www.elsevier.nl/locate/entcs

ends at the depot, each required demand is serviced by one single vehicle and the total weight of demands handled by any vehicle does not exceed vehicle capacity. Applications of the CARP and its extensions arises in several contexts in public services as road gritting ([8]), urban waste collection([6]), or street maintenance ([25], [30]). CARP has been shown to be NP-Hard([12]). Several heuristics and exact algorithms have been proposed for it. Heuristics were able to provide solutions at a low computational cost and the use of metaheuristics ([22]) techniques improved results obtained by traditional heuristics.

We developed an algorithm for CARP based on the ideas of the Biased Random Key Genetic Algorithms (BRKGA) proposed in ([9],[14]) and local search. BRKGA include improvements to genetic algorithms that are intended to avoid some problems that happen with classical chromosome representation and the Parameterized Uniform Crossover[28].

This paper is organized as follows: section 2 presents a mathematical model for CARP and a review of the literature. At section 3 Biased Random Key Genetic Algorithms (BRKGA) are described. Our BRKGA algorithm for CARP is presented at section 4. Computational experiments carried on in order to evaluate the performance of the method are presented in section 5. Finally, section 6 is devoted to concluding remarks and future work.

2 The Capacitated Arc Routing Problem

As we already mentioned, CARP offers interest because it is a difficult problem and at the same time it has several real world applications as waste collection, road maintenance, mail delivery, etc. Several organizations can benefit of having good solutions for CARP.

CARP may have additional characteristics and constraints as: unhomogeneous fleet of vehicles, time windows, mixed one way and two way streets, several depots, etc.. In these cases the model and algorithms presented here can be modified accordingly.

2.1 Mathematical Model

We describe here the mathematical model for CARP proposed by Maniezzo [24]. Let:

- G=(V,E) be a graph representing the roads network,
- V the set of vertex,
- E the set of arcs,
- R $\subseteq$ E the set of required arcs,
- $V_r \subseteq V$ the set of vertex incidents at arcs of R plus a node associated with

the depot,

- K={1,..., M} the fleet of vehicles, all of them with capacity Q,
- c_{ij} cost of arc (i,j) ∈ E,
- q_{ij} demand associated at arc (i,j) in R

$$q_{ij} = \begin{cases} 0 & \text{si (i,j)} \notin \text{R} \\ > 0 & \text{si (i,j)} \in \text{R} \end{cases}$$

Variables are defined as:

$$x_{ijk} = \begin{cases} 1 \text{ if arc (i,j) is traversed by vehicle k} \\ 0 \text{ otherwise} \end{cases}$$

CARP can be modeled as a zero-one integer linear problem as follows:

$$\text{Minimize} \sum_{(i,j)\in E} c_{ij} \sum_{k\in K} x_{ijk} \tag{1}$$

$$\sum_{k\in K} x_{ijk} = 1 \qquad (i,j) \in R \tag{2}$$

$$\sum_{j\in V_r - \{0\}} \sum_{k\in K} x_{0jk} = \mid K \mid \tag{3}$$

$$\sum_{(i,j)\in R} q_{ij} x_{ijk} \leq Q \qquad k \in K \tag{4}$$

$$\sum_{j\in V_r} x_{ijk} = \sum_{j\in V_r} x_{jik} \qquad i \in V_r, k \in K \tag{5}$$

$$\sum_{i\in S} \sum_{j\notin S} x_{ijk} \geq \sum_{j\in V} x_{hjk} \qquad S \subseteq V_r - \{0\}, k \in K, h \in S \tag{6}$$

$$x_{ijk} \in \{0,1\} \qquad (i,j) \in R, k \in K \tag{7}$$

The objective function (1) seeks to minimize the sum of the costs of traversed arcs, subject to the following constraints:

- all required arcs must be visited by exactly one vehicle (2).
- the number of vehicles can not exceed the total number of available vehicles (3).
- total load of any vehicle can not exceed Q (4).
- flow conservation constraints (5).
- subtour elimination constraints (6).

2.2 Previous Work

Several heuristics have been proposed for the CARP. Among them we can mention the Augment-Merge algorithm proposed by Golden and Wong [12], a Path Scanning algorithm presented by Golden et al. [11], a tour splitting algorithm from Ulusoy [29] and the Shorten-Switch-Add-Drop-Cut of Hertz and Mittaz. Also heuristics originally developed for similar problems were adapted to CARP, as for example 2-Opt that was applied by Lacomme et al. [23] and Beullens et al. [4]. Maniezzo [24] developed an heuristic based on 3-Opt and other improvement methods.

Several metaheuristics for CARP have been developed since 15 years ago. Eglese [8] developed a Simulating Annealing approach for a gritting problem. Hertz et al. [19] designed a Tabu Search algorithm, CARPET, based on the previously mentioned ADD and DROP algorithms, and an objective function with penalties. They had obtained very good results on instances of the literature. Also Brandão and Eglese [5] developed a Tabu Search method. Hertz and Mittaz present a Variable Neighborhood Descent method [20]. A tour visiting all the required arcs is generated, and then it is divided and improved using algorithms Shorten, Cut and Switch. Results on instances from DeArmon, Belenguer and other instances show that the algorithm obtains better results and in less computational time than CARPET. On their side, Beullens et al. presented in [4] a Guided Local Search. An initial solution is improved using 2-opt and other interchange between routes operators . They also report very good results.

In spite of CARP being NP-Hard, a few exact algorithms had been developed for it, that provide solutions for small and medium instances. Hirabayashi et al. [21] presented a branch-and-bound method. They obtained optimal solutions on instances that have 15 to 50 arcs. Belenguer and Benavent [2] developed a cutting plane algorithm that was tested on the instances of Benavent, Golden, Kiuchi y Eglese. The gap with the best known solution on the first three sets of instances with up to 50 nodes and 97 arcs, was less than 1%. On the instances of Eglese (up to 140 nodes and 190 arcs) this gap was less than 3%.

3 Biased Random Key Genetic Algorithm

3.1 Genetic Algorithms

Genetic Algorithms, GA, ([10]) are based on simulating the evolution of a population over a number of generations. They apply the concept of survival of the best fitted individuals to find optimal or near-optimal solutions to combinatorial optimization problems.

A *population* of solutions evolves over a number of iterations or *generations*

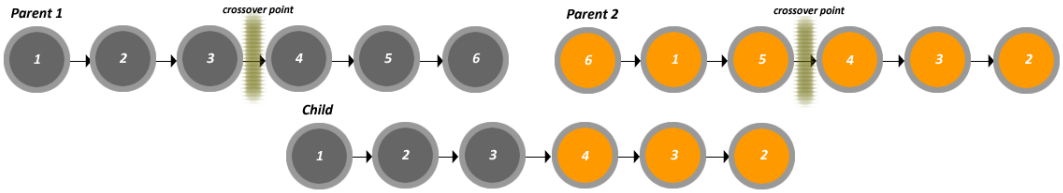

Fig. 1. GA: Infeasible solution after crossover

in order to explore the space of solutions and trying to avoid getting entrapped at local optima. Probabilistic transition rules are used. A solution of the optimization problem is called an *individual* or *chromosome*. Usually solutions are coded by a finite chain of bits or integers called *genes*. Each *gene* can take a value called an *allele* from a finite alphabet. This coding allows to represent the reproduction between parents. The objective function of the combinatorial optimization problem is used as fitness criteria to select good individuals.

At each iteration of a GA, that is at each generation, a new population is created by combining elements of the current population. Three operators are applied: reproduction, crossover, and mutation. That is, a small percentage of the best individuals in terms of fitness is directly copied to the next population. Next, crossover operators (deterministic or probabilistic) are applied to random selected parents to produce offspring for the next generation. Then, random mutation of some alleles is carried on in random selected individuals as a mean of escaping from local optima.

3.2 Random Key Genetic Algorithms

Random-Key Genetic algorithms, RKGA, were introduced by Bean [1] for solving sequencing problems. In RKGA chromosomes are represented as strings or vectors of randomly generated real numbers in the interval [0,1]. A deterministic algorithm, called a decoder, takes as input a vector of keys and associates it with a solution of the combinatorial optimization problem.

RKGA is an improvement of Genetic Algorithms, that is intended to overcome the difficulty appearing in GA when the offspring obtained by the crossover operator is not feasible for the original problem. The random key representation of solutions allows to define crossover and mutation operators that are independent of the particular problem we are dealing with. [27].

For example, assume that we have a TSP instance of 6 vertexes. If we assign numbers 1 to 6 to cities, a solution can be represented by a permuntation of vector (1,2,3,4,5,6) Lets consider the following tours:

Parent 1: $1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 6$

Parent 2: $6 \rightarrow 1 \rightarrow 5 \rightarrow 4 \rightarrow 3 \rightarrow 2$

Applying a one-point crossover operator to these chromosomes may produce an infeasible offspring, as it is shown at Fig. 1.

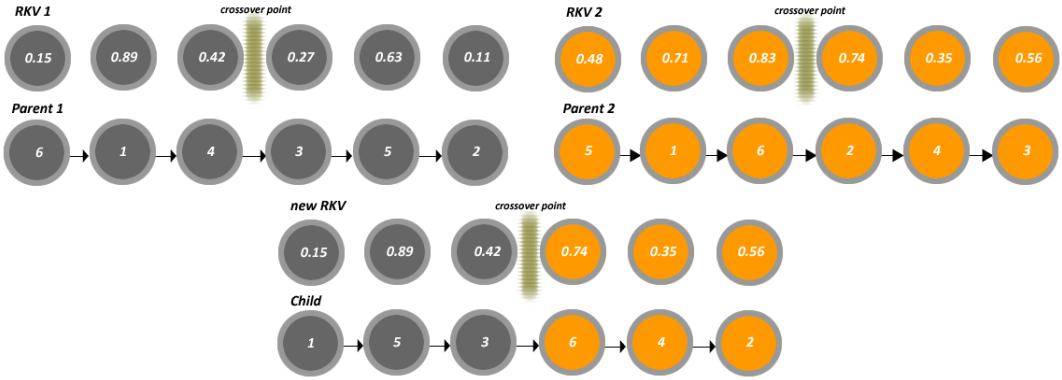

Fig. 2. RKGA: Feasible solution after crossover two random key vectors

To provide a RKGA representation of solutions for this problem, we generate chains of randomly generated real numbers on the [0,1] interval. For example we may have the following vectors:

RKV1: (0.15, 0.89, 0.42, 0.27, 0.63, 0.11)
RKV2: (0.48, 0.71, 0.83, 0.74, 0.35, 0.56)

To decode these chains as TSP solutions we sort them in an increasing order of the keys. Mapping the i-position in the chain with the position of the key in the ordered chain we get the corresponding TSP tour.

RKV1: (0.11, 0.15, 0.27, 0.42, 0.63, 0.89)
Parent1 : 6→1→4→3→5→2
RKV2: (0.35, 0.48, 0.56, 0.71, 0.74, 0.83)
Parent2: 5→1→6→2→4→3

After applying crossover and sorting the offspring, we get a new feasible solution for the TSP, as it is shown in Fig. 2.

In order to design a RKGA algorithm for a combinatorial optimization problem we need to randomly generate an initial population of random keys vectors. Then operators similar to those of Genetic Algorithms are applied at each generation as follows:

- After the fitness of each individual is computed by the decoder, the population is partitioned into two groups of individuals: a small group of p_e *Elite* solutions and the remaining set of $(1-p_e)$ *Non-Elite* individuals. The *Elite* individuals are copied without change in the population of next generation.
- Parameterized Uniform Crossover[28]: two parents from the entire population are ramdomly selected and crossed. Each allele of the offspring is randomly taken from one parent with a probability (p) or from the other with probality $(1-p)$.

  Fig. 3 shows how the crossover operator with p=0.7 produces a new individual which inherites more keys from the Elite parent than form the Non-Elite one.

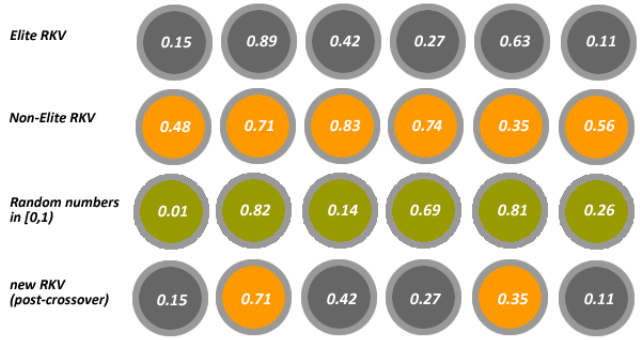

Fig. 3. RKGA: Crossover operation using two random key vectors

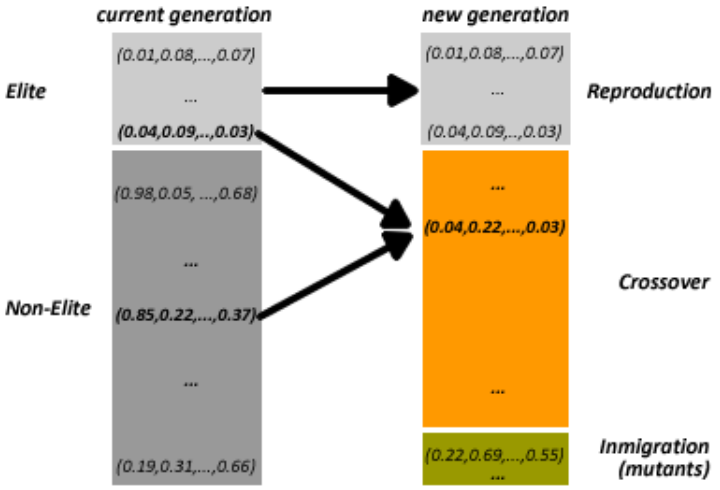

Fig. 4. RKGA: Transition between generations

- Mutation: RKGA implements mutation by introducing at each generation a small number of *mutants* into the population. A mutant is an array of random keys generated as it was done for the initial population.

This RKGA overall procedure is illustrated at Fig. 4.

3.3 BRKGA

Biased Random Key Genetic Algorithms were simultaneously proposed by Gonçalves and Almeida [14] and by Ericsson et al. [9] . They differs from RKGA in the way parents are selected for mating. In BRKGA each new individual is obtained combining one element selected at random from the Elite partition in the current population and the other from the Non-Elite set of individuals. Repetition in the selection of a mate is allowed. This way and also because the mechanism of implement mating is the Parameterized Uniform Crossover operator with p_e >0.5, the probability that an individual inherits the allele from the Elite parent is increased.

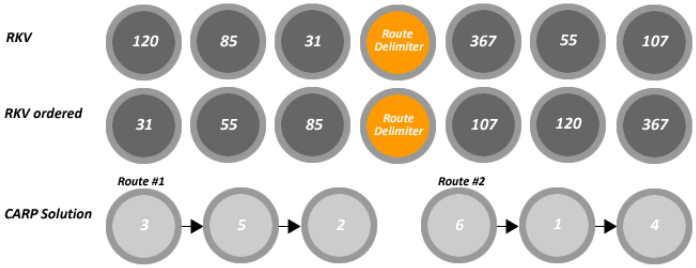

Fig. 5. BRKGA: Random keys encoding and decoding for CARP

BRKGA has been successfully applied to several combinatorial optimization problems as Packing [13], Scheduling, Resource Constrained Project Scheduling [17], Assembly Line Balancing [14],Manufacturing Cell Formation [15] and the iteSnyderBRKGAGTSP2006, [26]). A recent survey on BRKGA is [16]

4 BRKGA algorithm for the CARP

Our algorithm is based on the framework proposed by Gonçalves ([13]) and Gonçalves and Almeida ([14]), although it also includes new features such as mutation, a restart phase and a new random key encoding for the CARP.

4.1 Encoding

Encoding for the CARP follows ideas presented in Samanlioglu et al. [26]. The vectors of random keys get random integer values on the interval [MinU, MaxU]. The length of each vector is equal to number of required arcs of the instance. Route delimiters (RD) are added to these vectors to allow rapid detection of each vehicle route in the solution. In order to have a complete solution of the problem and to evaluate the fitness of it, the minimum cost path between the incident nodes of the successive required arcs at each of these routes has to be determined.

For example, lets assume that we have a CARP instance with 6 required arcs with every demand=1, vehicle capacity = 4 and MinU=50 and MaxU=200. Given the following random key vector:

RKV: 120 85 31 RD 367 55 107

if we sort it in ascending order and fix the route delimiters positions, the solution obtained is:

CARP solution (2 routes): 3→5→2,6→1→4

As shown in Fig. 5, this encoding is suitable for random keys crossover and allows solution improvement of the decoded solution by means of local search.

4.2 Initial population

Each individual of the initial population is generated as follows:

(i) Random key vector generation: Alleles are randomly generated within the range $[MinU, MaxU]$. Its size is equal to the number of required arcs.

(ii) Random key vector decoding: sorting the vector in increasing order or they values and looking for the value positions, as it has been already explained for the TSP, we obtain a tour where the capacity constraint is not taken into account.

(iii) Feasible CARP solution: applying the heuristic Iterated Tour Partitioning proposed by Haimovich and Kan[18], the tour obtained in the previous step is partitioned in subtours and a feasible solution is obtained. This procedure is applied several times starting at a diferent position each time. The best solution is kept.

(iv) CARP solution improvement: local search operators are applied.

(v) CARP solution encoding: given the initial random key vector and the improved CARP solution, a reverse mapping is made in order to obtain a random key vector with route delimiters.

At step iii) the vehicle capacity is multiplied by a real number between 0 and 1, which is modified along the execution of the algorithm. This produces initial solutions of diferent sizes and helps to have both inter and intra route local search methods of step iv) be used with similar frequency.

4.3 GA operators

Before applying genetic operators, all random key vectors from the previous generation are sorted according to their fitness. Then, they are classified as Elite individuals, Non-Elite individuals and random key vectors that will be mutated.

4.4 Crossover

Taking an Elite individual and a Non-Elite one, the Parameterized Uniform Crossover [28] operator is applied. The new individual inherits alleles from the first parent with probability p_e and with $(1-p_e)$ from the second one. The selection of parents is deterministic. Given that all individuals are previously classified, a parent (from both types) is selected in a sequential way. Therefore, it is possible that an Elite individual mates with several Non-Elite ones.

Each individual obtained this way is decoded and problem feasibility is checked. If it is not feasible some tours are divided in order to fulfill cacapity constraint. The decoded solution is then improved using local search operators and finally it is encoded again as a random key vector.

Fig. 6. BRKGA: Mutation over a random key vector

4.5 Mutation

BRKGA framework includes to generate new individuals (or mutants) in each generation ([14], [27], [26]). However, in our algorithm a classic GA mutation operator is applied to some individuals previously selected from the actual population. This operator is illustrated in Fig. 6 in a TSP example.

After having modified certain alleles (this quantity is variable) of an individual, the decoded CARP solution is improved. Then, applying reverse mapping, a new individual (random key vector) is obtained.

4.6 Local search

Given a new solution obtained as described above local search operators are applied. We followed ideas on Beullens et al. [4]. In order to improve the solution quality and to save computational time a mechanism of neighbor lists and arc marking was implemented. For each required arc e we create a list *neigh(e)* of required neighbor arcs. Then we search for an improved solution with the following procedure:

- Start: if the list of required arcs is empty then return "best solution"'. Otherwise, select the first arc e from the list and go to Next. All arcs in the list *neigh(e)* of neighbors of e are labeled as unexamined.
- Next: select the first neighbor in the neighbor list *neigh(e)* of arc e. If all arcs are examined, go to Unmark. Otherwise, determine if both arcs are on the same route (in the current solution). If there are, go to Improvement-1R, else go to Improvement-2R.
- Improvement-1R: determine if it is possible to improve current solution applying some intra-route operator (see Table 1). If it is possible, apply the move, mark arcs involved, save new solution and go to Start. Otherwise, the neighbor arc of e is examined. Go to Next.
- Improvement-2R: determine if it is possible to improve current solution applying inter-route operators following the steps mentioned in the previous

Table 1
BRKGA-Improvement: arc marking strategy

Position of e	Position of *neighbor(e)*	Move	Arc marking
i	j	2-Opt(i, i+1, j, j+1)	arc(i+1), arc(j), arc(j+1)
j+1	i+1	2-Opt(i, i+1, j, j+1)	arc(i), arc(i+1), arc(j)
i	j	Relocate(i-1, i, i+1, j, j+1)	arc(i-1), arc(i+1), arc(j), arc(j+1)
i	j	Relocate(i-1, i, i+1, j-1, j)	arc(i-1), arc(i+1), arc(j-1), arc(j)
i	j	Relocate(i, i+1, j-1, j, j+1)	arc(i+1), arc(j-1), arc(j), arc(j+1)
i	j	Relocate(i-1, i, j-1, j, j+1)	arc(i-1), arc(j-1), arc(j), arc(j+1)
i	j	Cross(i,i+1, j-1, j)	arc(i+1), arc(j-1), arc(j)
i	j	Cross(i-1, i, j,j+1)	arc(i-1), arc(j), arc(j+1)

step.

- Unmark: delete arc e from list and then go to Start.

This strategy is intended to intensify the search over arcs whose connections allow getting better solutions. If the arc marked after an improvement is not actually in the list, it is added (and its neighbors are labeled as *unexamined*).

In Table 1 moves analyzed in order to reach better solutions are detailed. Unattractive moves are avoided this way. 2-OPT was used as intra-route operator and Relocate and Cross were used for Inter-route improvement. See [4] for more details.

4.7 Restart

Generation of mutants in RKGA/BRKGA is a simple mechanism to reduce probability of getting trapped in local optima. We propose another alternative to do this. At every generation, classic mutation operator over individuals previously selected is applied. However, this action is not enough to skip poor solution regions.

So, if the best solution found is not improved during certain number of generations a new population is generated. Elite individuals are kept and new individuals are generated to complete the new population. Keeping elite individuals attempts to accelerate convergence to new locals optima.

5 Computational results

We report now the results obtained on a set of experiments conducted to evaluate the performance of our BRKGA algorithm. The algorithm was implemented within the Eclipse framework and the computational experiments were carried out on a PC Core 2 4300, 1.80Ghz, 2Gb RAM and Windows Vista operating system. It has been run over several instances from the literature. These were taken from Kiuchi et al. [21], Golden et al. [11] and Benavent et al. [3]. Table 2 shows data of these instances: number of vertexes, number of required arcs, demand, vehicle capacity, and best known solution including number of vehicles on it and total traveling cost. It also includes the maximum CPU time (in seconds) allowed for each run. Best solutions obtained by Kiuchi et al. instances are known to be optimal.

Based on the experience reported at the literature ([14], [26], [27]) and on previous computational tests, the parameters used by our algorithm are:

- Population size: 30 individuals
- Number of elite individuals: 20% of the entire population
- Number of mutated individuals: 20% of the entire population
- Crossover probability p_e: 70%
- Restart time: CPU time/4

For each instance, 50 runs were performed. In table 3, results obtained by our BRKGA algorithm are listed. The table includes for each instance: name, average total traveling cost, standard deviation, mode, minimum total cost, occurrence of this value (%), maximum total cost and average CPU time (in seconds).

Some conclusions can be derived from the results:

- Optimal or near optimal total traveling cost matches the best known results in most instances (21 of 25). With respect to the others, their gaps are less than 4% (Gdb22 2%, Val8A 0.2%, Val8B 3.5% and Val9A 4%).
- At 15 instances the best result is reached in more than 25% of the runs. In 5 of the instances (Kshs1, Kshs2, Kshs6, Gdb15 and Gdb17) 100% of effectiveness was obtained.
- The average total traveling cost is less than 10.7% away from the best result. In 19 instances, this gap is less than 5%.
- The mode matches the best result in 15 out of 25 instances. In 5 of the remaining ones (Gdb16, Gdb18, Gdb22, Val6A and Val8A) this gap is lower than 5%.
- The range between maximum and minimum total traveling cost obtained by different runs on the same instance is less than 5% of the best known

Table 2
Instances information

Instance	$\|V\|$	$\|R\|$	Demand	Q	Best Solution		CPU Time(s.)
					TD	Veh.	
Kshs1	8	15	535	150	14661	4	15
Kshs2	10	15	497	150	9863	4	15
Kshs3	6	15	565	150	9320	4	15
Kshs4	8	15	594	150	11498	4	15
Kshs5	8	15	443	150	10957	3	15
Kshs6	9	15	389	150	10197	3	20
Gdb1	12	22	22	5	316	5	30
Gdb3	12	22	22	5	275	5	20
Gdb10	12	25	37	10	275	4	20
Gdb14	7	21	89	21	100	5	20
Gdb15	7	21	112	37	58	4	20
Gdb16	8	28	116	24	127	5	45
Gdb17	8	28	168	41	91	5	25
Gdb18	9	36	153	37	164	5	90
Gdb20	11	22	107	27	121	4	20
Gdb21	11	33	154	27	156	6	30
Gdb22	11	44	205	27	200	8	60
Val1A	24	39	358	200	173	2	90
Val2A	24	34	310	180	227	2	120
Val3A	24	35	137	80	81	2	120
Val3B	24	35	137	50	87	3	90
Val6A	31	50	451	170	223	3	360
Val8A	30	63	566	200	386	3	600
Val8B	30	63	566	150	395	4	600
Val9A	50	92	654	235	323	3	1800

Table 3
Computational Results

Instance	Results						
	$\bar{TD}$	DE	MO	$Min.TD$	$\%Min.TD$	$Max.TD$	$\bar{Time}$
Kshs1	14661	0.0	14661	14661	100	14661	3.6
Kshs2	9863	0.0	9863	9863	100	9863	0.6
Kshs3	9323.48	17.22	9320	9320	96	9407	3.4
Kshs4	12310.7	818.49	11498	11498	18	15767	8.6
Kshs5	10994.04	99.15	10957	10957	86	11299	4.5
Kshs6	10197.00	0.0	10197	10197	100	10197	0.7
Gdb1	316.46	1.48	316	316	90	323	12.7
Gdb3	277.12	2.30	275	275	52	281	8.7
Gdb10	278.52	3.86	275	275	42	287	9.9
Gdb14	100.36	0.77	100	100	82	102	6.7
Gdb15	58	0.0	58	58	100	58	0.6
Gdb16	128.80	0.60	129	127	10	129	6.7
Gdb17	91.00	0.0	91	91	100	91	1.6
Gdb18	165.08	1.00	166	164	46	166	32.5
Gdb20	134.02	9.50	129	121	6	160	13.3
Gdb21	168.58	4.72	172	156	2	178	25.9
Gdb22	211.24	4.12	209	204	2	226	54.4
Val1A	173.14	0.60	173	173	92	177	35.5
Val2A	227.56	1.21	227	227	76	233	51.6
Val3A	81.58	0.67	81	81	52	83	43.5
Val3B	91.38	1.56	92	87	2	95	41.7
Val6A	229.46	2.91	229	223	2	237	190.9
Val8A	402.22	6.10	403	387	2	414	317.6
Val8B	428.34	6.06	430	409	2	440	259.5
Val9A	340.12	2.22	340	336	4	344	884.3

value in 17 of them. In 4 of the other instances this range is less than 10%.

- Computational times were less than 60% of the maximum allowed time in 22 instances (exceptions were Gdb20, Gdb21 and Gdb22).

6 Conclusions and future work

This paper presents an algorithm for CARP based on BRKGA metaheuristic and local search. Computational results on instances from the literature have shown that this is an effective heuristic, competitive with the best algorithms reported so far. Optimal or near optimal solutions were obtained in a robust way using low CPU effort. This is due to several facts. The way the initial population was generated allowed intensification of the searching in different regions. The Parameterized Uniform Crossover operator employed to mate Elite with a NonElite individuals and the improvement phase after crossover and mutation showed to be very effective. The neighbor list and classical local search methods permitted intensification of the search process in attractive regions.

Anyway there is still place to improve the algorithm. The initial population could be generated using other heuristics to better explore the solution space. Fitness is a simple way to measure quality of solutions. However, a mechanism to analyze diversity of solutions could be considered. This would allow exploring different solution regions in a systematic way.

References

[1] Bean, J., *Genetic algorithms and random keys for sequencing and optimization*, ORSA Journal on Computing **6** (1994), 154–160.

[2] Belenguer, J., and E. Benavent, *A Cutting Plane algorithm for the capacitated arc routing problem*, Computers and Operations Research **30** (2003), 705–720.

[3] Benavent, E., V. Campos, A. Corberán, and E. Mota, *The capacitated arc routing problem: lower bounds*, Networks **22** (1992), 669–690.

[4] Beullens, P., L. Muyldermans, D. Cattrysse, and D. Van Oudheusden, *A Guided Local Search heuristic for the capacitated arc routing problem*, European Journal of Operational Research-Discrete Optimization **147** (2003), 629–643.

[5] Brandão, J., and R. Eglese, *A Deterministic Tabu Search Algorithm for the Capacitated Arc Routing Problem (CARP)*, Computers and Operations Research **35-4** (2008), 1112–1126.

[6] Del Pia, A., and C. Filippi, *A Variable Neighborhood Descent algorithm for a real waste collection problem with mobile depots*, International transactions in Operational Research **13** (2006), 125–141.

[7] Dorigo, M., and T. Stützle, "Ant Colony Optimization" 1st Ed., MIT Press, Massachusetts, 2004.

[8] Eglese, R., *Routing winter gritting vehicles*, Discrete applied mathematics **48-3** (1994), 231–244.

[9] Ericsson, M., M. Resende, and P. Pardalos, *A genetic algorithm for the weight setting problem in OSPF routing*, Journal of Combinatorial Optimization **6** (2002), 299–333.

[10] Goldberg, D., "Genetic algorithms in search, optimization and machine learning" 1st Ed., Addison-Wesley, Massachusetts, 1989.

[11] Golden, B., J. DeArmon, and E. Baker, *Computational experiments with algorithms for a class of routing problems*, Computers and Operations Research **10-1** (1983), 47–59.

[12] Golden, B., and R. Wong, *Capacitated arc routing problems*, Networks **11** (1981), 305–315.

[13] Gonçalves, J. F., *A hybrid genetic algorithm-heuristic for a two-dimensional orthogonal packing problem*, European Journal of Operational Research **183** (2007), 1212–1229.

[14] Gonçalves, J. F., and J. Almeida, *A hybrid genetic algorithm for assembly line balancing*, Journal of Heuristics **8** (2002), 629–642.

[15] Gonçalves, J. F. and M. G.C. Resende, *An evolutionary algorithm for manufacturing cell formation* , Computers and Industrial Engineering, **47**(2004), 247–273.

[16] Gonçalves, J. F. and M. G.C. Resende, *Biased random-key genetic algorithms for combinatorial optimization*, Journal of Heuristics **17** (2011), 487–525.

[17] Gonçalves, J. F., M. Resende, and J. Almeida, *A biased random-key genetic algorithm with forward-backward improvement for the resource constrained project scheduling problem*, Journal of Heuristics **published Online** (2010).

[18] Haimovich, M., and A. Kan, *Bounds and heuristics for capacitated routing problems*, Mathematics of Operations Research **10** (1985), 527–542.

[19] Hertz, A., G. Laporte and M. Mittaz, *A Tabu Search heuristic for the capacitated arc routing problem*, Operations Research **48** (2000), 129–135.

[20] Hertz, A., and M. Mittaz, *A Variable Neighborhood Descent Algorithm for the Undirected Capacitated Arc Routing Problem*, Transportation Science **35-4** (2001), 425–434.

[21] Hirabayashi, R., Y. Saruwatari, and N. Nishida, *Tour construction algorithm for the capacitated arc routing problem*, Asia-Pacific Journal of Operational Research **9** (1992), 155–175.

[22] Hoos, H., and T. Stützle "Stochastic Local Search Foundations and Applications," 1st Ed., McGraw-Hill, San Francisco, 2005.

[23] Lacomme, P., C. Prins, and A. Tanguy, *First Competitive Ant Colony Scheme for the CARP*, Lecture Notes in Computer Science **3172** (2004), 426–427.

[24] Maniezzo, V., *Algorithms for large directed CARP instances: urban solid waste collection operational support*, University of Bologna, Department of Computer Science, Technical Report UBLCS-2004-16.

[25] Muydelrmans, L., D. Cattrysse, D. Van Oudheusden, and T. Lotan, *Districting for salt spreading operations*, European Journal of Operational Research **139** (2002), 521–532.

[26] Samanlioglu, F., W. Ferrell, and M. Kurz, *A memetic random-key genetic algorithm for a symmetric multi-objective salesman problem*, Computers and Industrial Engineering **55** (2008), 439–449.

[27] Snyder, L., and M. Daskin, *A random key genetic algorithm for the generalized traveling salesman problem*, European Journal of Operational Research **174** (2006), 38–53.

[28] Spears, W., and K. DeJong, *On the Virtues of Parameterized Uniform Crossover*, Proceedings of the Fourth International Conference on Genetic Algorithms, San Diego-USA (1991), 230–236.

[29] Ulusoy, G., *The fleet size and mixed problem for capacitated arc routing*, European Journal of Operational Research **22** (1985), 329–337.

[30] Zhu, Z., X. Li, Y. Yang, X. Deng, M. Xia, and Z. Xie, *A hybrid genetic algorithm for the multiple depot capacitated arc routing problem*, Proceedings of the IEEE International Conference on Automation and Logistics, Jinan-China (2007), 2252–2258.