Applied Mathematical Modelling 36 (2012) 2110–2117

Contents lists available at SciVerse ScienceDirect

Applied Mathematical Modelling

journal homepage: www.elsevier.com/locate/apm

Optimised crossover genetic algorithm for capacitated vehicle routing problem

Habibeh Nazif [a,*], Lai Soon Lee [b]

[a] Department of Mathematics, Faculty of Science, Universiti Putra Malaysia, 43400 UPM Serdang, Selangor, Malaysia
[b] Laboratory of Applied and Computational Statistics, Institute for Mathematical Research, Universiti Putra Malaysia, 43400 UPM Serdang, Selangor, Malaysia

ARTICLE INFO

Article history:
Received 10 June 2010
Received in revised form 24 July 2011
Accepted 9 August 2011
Available online 19 August 2011

Keywords:
Genetic algorithm
Vehicle routing problem
Combinatorial optimisation
Network
Heuristics

ABSTRACT

This paper presents a genetic algorithm for solving capacitated vehicle routing problem, which is mainly characterised by using vehicles of the same capacity based at a central depot that will be optimally routed to supply customers with known demands. The proposed algorithm uses an optimised crossover operator designed by a complete undirected bipartite graph to find an optimal set of delivery routes satisfying the requirements and giving minimal total cost. We tested our algorithm with benchmark instances and compared it with some other heuristics in the literature. Computational results showed that the proposed algorithm is competitive in terms of the quality of the solutions found.

1. Introduction

The vehicle routing problem (VRP) is a hard combinatorial optimisation problem with numerous industrial applications. In this study, we consider the capacitated vehicle routing problem (CVRP) which is an extention of VRPs. The CVRP is formally defined as an undirected graph $G = (V, E)$ where $V = \{v_0, v_1, \ldots, v_n\}$ is a vertex set and $E = \{(v_i, v_j) | v_i, v_j \in V, i < j\}$ is an edge set. The depot is represented by vertex v_0, which uses m independent delivery vehicles, with identical delivery capacity Q, to service demands q_i from n cities or customers, $i = 1,2,\ldots,n$, represented by the set of n vertices $\{v_1, \ldots, v_n\}$. Each customer v_i has a service or drop time t_i. A non-negative cost (distance or travel time) matrix $C = (c_{ij})$ between customers v_i and v_j is defined on E. A solution for the CVRP would be a partition $R_1, R_2, \ldots, R_m$ of V representing the routes of the vehicles, each route $R_i = \{v_{i0}, v_{i1}, \ldots, v_{ik+1}\}$, where $v_{ij} \in V$ and $v_{i0} = v_{ik+1} = 0$ (0 denotes the depot), satisfying $\sum_{v_{ij} \in R_i} q_j \leqslant Q$. The cost of the problem solution is the sum of the costs of its routes R_i, defined as follows:

$$Cost = \sum_{i=1}^{m} Cost(R_i) = \sum_{j=0}^{k} c_{jj+1} + \sum_{j=0}^{n} t_j. \tag{1.1}$$

The CVRP consists in determining a set of a maximum of m routes of minimum total cost, such that each route starts and ends at the depot, each customer is visited exactly once by exactly one vehicle, subject to the restriction that the total demand of any route does not exceed Q. We additionally consider that the total duration of any route is not longer than a preset bound D.

A survey on the CVRP and variants is given by Toth and Vigo [1] and an overview of heuristics and metaheuristics may also be found in Laporte et al. [2] and Cordeau et al. [3,4]. The tabu search implementations of Taillard [5] and Rochat and

* Corresponding author.
E-mail addresses: habibeh@math.upm.edu.my (H. Nazif), lee@math.upm.edu.my (L.S. Lee).

0307-904X/$ - see front matter © 2011 Elsevier Inc. All rights reserved.
doi:10.1016/j.apm.2011.08.010

Taillard [6] have obtained the best known results to benchmark VRPs. Osman [7], Gendreau et al. [8], Rego and Roucairol [9], Barbarosoglu and Ozgur [10], and Toth and Vigo [11] have reported similar results obtained using tabu search. Osamn [7] and Hiquebran et al. [12] also achieved similar results by applying simulated annealing to the problem. However, Renaud et al. [13] observed that such heuristics required substantial computing times and several parameter settings. Bullnheimer et al. [14] and Gambardella et al. [15] proposed ant colony for VRPs. Their approaches have given results which are only slightly inferior to those from tabu search.

Mester and Bräysy [16] presented an adaptation of the active guided evolution strategies metaheuristic for the CVRP. The results demonstrated that their suggested method is highly competitive. Tarantilis [17] developed an adaptive memory programming method for solving the CVPR. Moreover, Particle Swarm Optimisation (PSO), which is a population based search method that mimics the behavior of group organism as a searching method, is proposed for solving the CVRP by Chen et al. [18] and Ai and Kachitvichyanukul [19,20].

Genetic algorithms (GAs) have also been applied for the VRPs. Potvin et al. [21] applied a hybrid approach to vehicle routing using neural networks and GA. Two hybrid GAs for the CVRP and distance VRP are proposed by Berger and Barkaoui [22] and Prins [23] respectively. Baker and Ayechew [24] considered the application of a GA to the basic VRP, which is competitive with other modern heuristics in terms of computing time and solution quality. Alba and Dorronsoro [25] proposed the utilization of some cellular GAs with and without including local search techniques for solving the CVRP. Also, Alba and Dorronsoro [26] focused in providing new optimal values for three additional benchmarks of Van Breedam [27], Golden et al. [28], and Taillard [5]. Their Computational results gave a high performance in terms of the quality of the solutions found and the number of function evaluations.

In this study, we propose an optimised crossover genetic algorithm (OCGA) for the CVRP. Section 2 gives a detailed descriptions of the proposed algorithm. In Section 3, the computational results are presented and discussed. In the final section, we give a brief conclusion.

2. Optimised crossover genetic algorithm (OCGA)

GAs were first proposed by John Holland [29] in the 1960s. The idea behind a GA is to model the natural evolution by using genetic inheritance together with Darwin's theory. A population of individuals representing tentative solutions is maintained over many generations. New individuals are produced by combining members of the population via crossover and mutation operators, and these replace existing individuals with some policies.

Most of the current methods of crossover determine a child by using a stochastic approach and without reference to the objective function. Optimised crossover was proposed by Aggarwal et al. [30] for the independent set problem. They applied optimised crossover within GA, which takes into account the objective function in a straightforward way, and produced two new children: the O-child (Optimum child) and E-child (Exploratory child). The O-child is constructed in such a way that has the best objective function value from a feasible set of children, while the E-child is constructed so as to maintain the diversity of the search space. Balas and Niehaus [31] developed this approach to produce a superior GA.

In the remainder of this section, we describe the proposed optimised crossover and discuss how various steps of the GA are implemented for the proposed algorithm. Briefly, the OCGA selects two parents from the population and uses an optimised crossover mechanism designed by a complete undirected bipartite graph to generate two children. The OCGA uses a swap node operator to produce children when the optimised crossover is not applied to the parents. To maintain the diversity within the population, two different mutations are randomly applied to each child. Moreover an elitism replacement scheme with filtration strategy is used to preserve good solutions and to avoid premature convergence. The general framework of OCGA can be shown as follows:

Algorithm. OCGA

begin

Initialise Population (randomly generated);
Fitness Evaluation;
repeat
Selection (probabilistic binary tournament selection);
Optimised Crossover;
Swap Node (if the optimised crossover is not applied);
Mutation (inversion and swap sequence);
Fitness Evaluation;
Elitism replacement with **Filtration**;
until the end condition is satisfied;
return the fittest solution found;
end

2.1. Gene representation and selection mechanism

The representation of a solution we use here is an integer string of length n, applied by Tan et al. [32]. Each gene in the string is the integer node number assigned to that customer originally. The sequence of the genes in the string is the order of visiting these customers. For example, if we have the following solution represented in Fig. 1:

Route No. 1: $0 \rightarrow 2 \rightarrow 3 \rightarrow 1 \rightarrow 4 \rightarrow 0$
Route No. 2: $0 \rightarrow 5 \rightarrow 6 \rightarrow 10 \rightarrow 0$
Route No. 3: $0 \rightarrow 8 \rightarrow 9 \rightarrow 7 \rightarrow 11 \rightarrow 12 \rightarrow 0$

The coded integer string is then

$$2 \rightarrow 3 \rightarrow 1 \rightarrow 4 \rightarrow 5 \rightarrow 6 \rightarrow 10 \rightarrow 8 \rightarrow 9 \rightarrow 7 \rightarrow 11 \rightarrow 12.$$

This representation is unique, and one string can only be decoded to one solution. Note we link the last customer visited in route i with the first customer visited in route $i + 1$ to form one string of all the routes involved, but we do not put any bit in the string to indicate the end of a route now, because such delimiters in a string greatly restrain the validity of children produced by optimised crossover operator later. To decode the string into route configurations, we simply insert the gene values into new routes sequentially. There is a chance that we may not get back the original routes after decoding, but it is generally assumed that minimising the number of routes helps in minimising the total travel cost, therefore, packing a route to its maximum capability implies a potential good solution as a result (see Tan et al. [32]).

After choosing the representation, we uniformly randomly generate an initial population using a random number generator. The population size affects the performance of GA as well as affecting the convergence rate and the running time. A population size between 30 and 200 is usually recommended in researches. In our study, the three different size 50, 100 and 150 are applied depending on the number of customers in the different problem instances.

Moreover we use a probabilistic binary tournament selection scheme to select individuals from the population to be the parents for the OCGA with a given selection probability $p_s = 0.75$. In other words, we give a 75% chance for the fitter individual to be selected as the parent compared to the less fit individual which only has a 25% chance to be selected.

2.2. Optimised crossover

We propose an optimised crossover operator within a GA for the CVRP. The proposed optimised crossover using a complete undirected bipartite graph finds the two new children which are called O-child and E-child. We will now explain the optimised crossover strategy on determining the O-child and E-child for the CVRP.

Step 1: Given two parents P_1 and P_2. Construct a undirected bipartite graph $G = (U \cup V, E_1 \cup E_2)$ where $U = \{u_1, u_2, \ldots, u_n\}$ representing customers, $V = \{v_1, v_2, \ldots, v_n\}$ representing nodes, E_1 and E_2 representing the arc sets in the graph in which, $\{u_j, v_i\} \in E_1$ if, and only if, customers j of parent P_1 is located at node i, and $\{u_j, v_i\} \in E_2$ if, and only if, customers j of parent P_2 is located at node i.

Step 2: Determine all perfect matchings in graph G. Suppose that there are k cycles corresponding to different locations of customers in the two parents. There will be exactly 2^k perfect matchings in graph G. Note that each perfect matching representing a temporary offspring.

Step 3: Select a temporary offspring with the least objective function value as O-child.

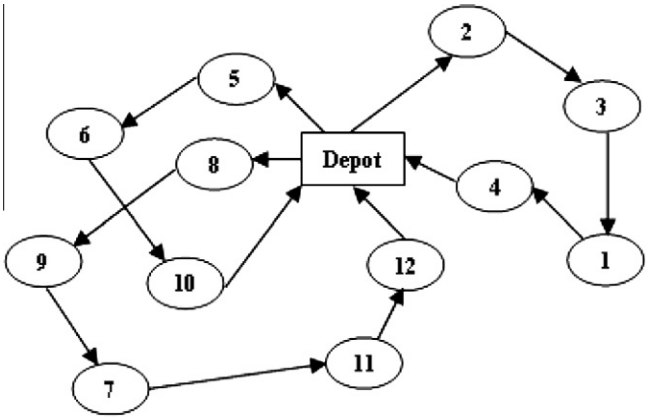

Fig. 1. A solution to a vehicle routing problem.

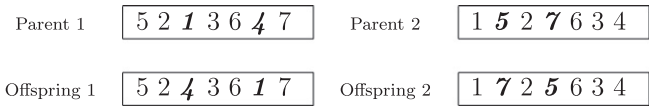

Fig. 2. An example of swap node.

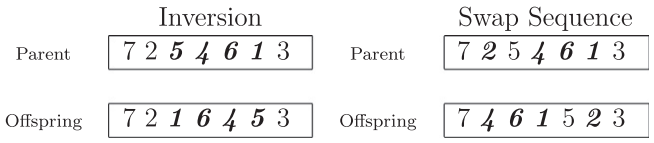

Fig. 3. An example of two mutation operators.

Step 4: Generate E-child using $((P_1 \cup P_2) \backslash O - child) \cup (P_1 \cap P_2)$.

We restricted the temporary offspring in graph G in every case to 2^5 even if the cycles are more than 5. In addition, the proposed optimised crossover is applied based on a crossover probability, $p_c = 0.75$.

2.3. Swap node

Crossover operator is usually applied to the selected parents with a crossover probability p_c, in which using a $0.60 < p_c < 0.95$ the better results will be achieved. Reproduction strategy is used when crossover is not applied to the selected parents. During this strategy two offspring are produced simply by duplicating the selected parents. In this study, the proposed optimised crossover is applied based on a crossover probability, $p_c = 0.75$, and a swap node operator instead of reproduction is used to produce two new offspring when the optimised crossover is not applied to the parents.

This operator could be regarded as a giant mutation where the elements in the parent are randomly reassigned. The process of swap node operator is as follows: two nodes from a parent are randomly selected and swaped. It is repeated for the second parent to create a second offspring.

An example of swap node operator for the two parents with 7 customers is given in Fig. 2. Nodes '3' and '6' from parent 1, representing customers '1' and '4' respectively, are randomly chosen and swaped to form offspring 1. The other nodes will remain unchanged. Similarly, offspring 2 is formed from parent 2.

2.4. Mutation

The mutation operator used in the proposed algorithm consists of applying with equal probability two different mutation operators called inversion and swap sequence operators. The inversion operator reverses the visiting order of the customers between two randomly selected points, while the swap sequence operator consists of randomly selecting two sub-strings of customers and exchanging them. The example of the two mutation operators are given in Fig. 3.

2.5. Elitism replacement with filtration

Elitism replacement scheme is applied in this study, as follows: both parent and offspring population are combined into a single population and sorted in a non-increasing order of their associated fitness value. Then, the first half of the combined population is selected as the individuals of the new population for the next generation. We propose the filtration strategy to identify identical individuals from the new population. The identical individuals will be removed and replaced by uniformly randomly generated new individuals to avoid premature convergence and to add diversity to the new population. As the filtration strategy requires a certain amount of computational time, the procedure will only be invoked after every 50 generations.

3. Computational experiments

In this section, we present the results of the computational experiments of our proposed algorithm on CVRP. The proposed algorithm is coded in C language and implemented on a Pentium 4, 2.0 GHz computer with 2.0 GB RAM.

The OCGA has been tested with problem instances from the benchmarks of Christofides et al. [33] and Taillard [5]. All the studied instances are publicly available at [34]. In the case of the benchmark of Christofides et al. [33], each instance contains between 50 and 199 customers as well as the depot. The instances with only capacity restriction on vehicles are C1–C5, C11 and C12. The others are with the additional restrictions of maximal route distance and the existence of a drop time.

Table 1
Comparison between crossover operators. The bold values mean the best values that are found by the operators.

Instance	Optimised crossover	1-point crossover
C1	**525.97**	527.14
C4	**1031.06**	1034.56
C8	**867.42**	869.07
C12	**820.88**	822.64

The instances of Taillard [5] are composed of 13 nonuniform problems of 75, 100, 150 and 385 customers, 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.

3.1. Initial investigations of OCGA

Initial investigations are performed to design the proposed OCGA. We gradually construct the proposed algorithm from a standard GA. The differences between the OCGA and SGA are with regards to the use of the crossover operator, reproduction procedure and the replacement scheme. For the initial investigation, we use instances C1, C4, C8, C12 from the benchmarks of Christofides et al. [33]. The numerical results are computed after making 50 independent runs for statistical significance, in which each run is terminated if the fitness is not improved after 100 generations.

We compare our proposed optimised crossover operator with the classic 1-point crossover operator to determine whether the proposed optimised crossover operator is advantageous to produce offspring. For this reason, the two crossover operators mentioned are employed in the SGA. The computational results of the comparison of the optimised crossover operator with the classic 1-point crossover operator are shown in Table 1.

From Table 1, we can clearly observe that the better solution quality is obtained under the optimised crossover operator. As a result, the optimised crossover operator is advantageous in generating better offspring.

The swap node operator is compared with the reproduction to investigate the effect on the solution quality when the crossover operator is not applied. The computational results of the swap node operator employed in the SGA which utilises the classic 1-point crossover operator, compared to the reproduction are reported in Table 2.

It is clear from Table 2 that the swap node operator reports better results compared to the reproduction. In fact, the swap node operator leads the search into the more interesting regions to explore better local optima by creating more diversity in population.

A comparison of the different replacement strategies employed in the SGA is given in Table 3. The steady-state replacement strategy with the proposed elitism replacement and filtration strategy are compared to assess the effectiveness of each replacement strategy on the solution quality.

From Table 3, we can conclude that the elitism replacement and filtration strategy outperforms the steady-state replacement strategy. Consequently, the solution space can be searched in a more efficient manner by using the elitism replacement and filtration strategy.

The demonstrated computational experiments provide guidelines to design the proposed OCGA. Thus, we apply the optimised crossover and the swap node operator to produce offspring in the proposed OCGA. The elitism replacement with filtration strategy is used to preserve good solutions and to avoid premature convergence.

Table 2
Results of swap node operator. The bold values mean the best values that are found by the operators.

Instance	Swap node	Reproduction
C1	**526.32**	529.21
C4	**1034.14**	1037.36
C8	**868.57**	870.44
C12	**821.75**	823.09

Table 3
Comparison between replacement strategies. The bold values mean the best values that are found by the operators.

Instance	Elitism with filtration	Steady state
C1	**526.89**	528.65
C4	**1035.27**	1038.04
C8	**869.62**	870.86
C12	**822.13**	824.50

Table 4
Computational results for the benchmark of Christofides et al. [33]. The bold values mean the best-so-far results that are found by the algorithms.

Instance	TS	SA	AGES	SEPAS	GA_P	GA_BB	OCGA	PD	BSF
C1	**524.61**	**524.61**	**524.61**	**524.61**	**524.61**	**524.61**	**524.61**	0.0000	**524.61**[a]
C2	838.60	840.61	**835.26**	**835.26**	**835.26**	**835.26**	**835.26**	0.0000	**835.26**[a]
C3	828.56	828.21	**826.14**	**826.14**	**826.14**	827.39	**826.14**	0.0000	**826.14**[a]
C4	1033.21	1037.57	**1028.42**	**1028.42**	1030.46	1036.16	**1028.42**	0.0000	**1028.42**[a]
C5	1318.25	1306.91	**1291.29**	1311.48	1296.39	1324.06	1299.64	0.6466	**1291.29**[b]
C6	**555.43**	**555.43**	**555.43**	**555.43**	**555.43**	**555.43**	**555.43**	0.0000	**555.43**[a]
C7	920.72	917.50	**909.68**	**909.68**	**909.68**	**909.68**	**909.68**	0.0000	**909.68**[a]
C8	869.48	**865.94**	**865.94**	**865.94**	**865.94**	868.32	**865.94**	0.0000	**865.94**[a]
C9	1173.12	1173.94	**1162.55**	**1162.55**	**1162.55**	1169.15	1163.38	0.0713	**1162.55**[a]
C10	1435.74	1415.53	1401.12	1407.21	1402.75	1418.79	1406.23	0.7436	**1395.85**[c]
C11	1042.87	1043.46	**1042.11**	**1042.11**	**1042.11**	1043.11	**1042.11**	0.0000	**1042.11**[a]
C12	**819.56**	**819.56**	**819.56**	**819.56**	**819.56**	**819.56**	**819.56**	0.0000	**819.56**[a]
C13	1545.51	1546.84	**1541.14**	1544.01	1542.86	1553.12	1542.25	0.0720	**1541.14**[a]
C14	**866.37**	**866.37**	**866.37**	**866.37**	**866.37**	**866.37**	**866.37**	0.0000	**866.37**[a]

[a] Taillard [5].
[b] Mester and Bräysy [36].
[c] Rochat and Taillard [6]

3.2. Results and discussions

We compare our proposed algorithm against the best-so-far (BSF) results reported in the literature. Concerning the instances of Christofides et al. [33], we have selected some algorithms namely: tabu search (TS) by Toth and Vigo [11], savings ants (SA) by Reimann et al. [35], metaheuristic called (AGES) by Mester and Bräysy [16], adaptive memory programming method called (SEPAS) by Tarantilis [17], and GAs that are those of Prins (GA_P) [23] and Berger and Barkaoui (GA_BB) [22].

In the case of the benchmark of Taillard [5], we selected an active guided evolution strategies metaheuristic called (AGES) proposed by Mester and Bräysy [36] and a cellular genetic algorithm called (JCell2o1i) by Alba and Dorronsoro [26]. We compute the percentage deviation (PD) between our best solution (sol) and the best-so-far (BSF) with the following equation:

$$PD = \left[\frac{sol - BSF}{BSF}\right] \times 100\%. \tag{3.1}$$

The numerical results are computed after making 50 independent runs for statistical significance, in which each run is terminated if the fitness is not improved after 100 generations. The proposed algorithm can easily be compared by consulting Tables 4 and 6, containing the best solution found with the instances of Christofides et al. [33], and the best solution found with the benchmark of Taillard [5], respectively.

It is clear from Table 4 that the OCGA finds the best-so-far solution for most of the instances described by Christofides et al. [33] earlier. As it can be seen, the deviation between our best solution and the best-so-far is always under 0.7436%.

The computational results from other researchers shown in Tables 4 and 6 are copied from their published papers where no computational times are reported, with the exception of the GA_P and the GA_BB for some instances.

Times are reported in Table 5 where the GA_P is implemented in Delphi on a 500 MHz PC under Windows 95; however, the GA_BB is implemented in C++ on a 400 MHz Pentium processor. Moreover, OCGA is implemented in C on a 2.0 GHz PC with 2.0 GB RAM.

Table 5 shows that the best time for instances C2 and C12 are held by GA_P, while GA_BB is faster for instances C8 and C14, and eventually, our OCGA is faster than the others for instances C1, C3, C6 and C7.

As shown in Table 6, in 5 cases of Taillard's instances, the best-so-far solution can be found by our algorithm. In other cases, the deviation between the best-so-far reported and the solution found by the OCGA is very low. In addition, we have

Table 5
Average time for the benchmarks of Christofides et al. (in seconds). The bold values mean the best times that are found by the algorithms.

Instance	GA_P	GA_BB	OCGA
C1	-	120	**48.33**
C2	**184**	860	235.21
C3	497	1674	**480.59**
C6	106	140	**75.09**
C7	315	630	**257.41**
C8	664	**303**	440.77
C12	**7**	433	267.13
C14	591	**284**	394.38

Table 6
Computational results for the benchmark of Taillard [5]. The bold values mean the best-so-far results that are found by the algorithms.

Instance	AGES	JCell2o1i	OCGA	PD	BSF
Tai75a	**1618.36**	**1618.36**	**1618.36**	0.0000	**1618.36**[a]
Tai75b	1344.64	**1344.62**	1344.63	0.0007	**1344.62**[b]
Tai75c	**1291.01**	**1291.01**	**1291.01**	0.0000	**1291.01**[a]
Tai75d	**1365.42**	**1365.42**	**1365.42**	0.0000	**1365.42**[a]
Tai100a	**2041.34**	2047.90	2050.64	0.4555	**2041.34**[c]
Tai100b	**1939.90**	1940.36	**1939.90**	0.0000	**1939.90**[d]
Tai100c	**1406.20**	1411.66	1408.40	0.1564	**1406.20**[c]
Tai100d	1581.25	1584.20	**1581.22**	0.0018	1581.25[c]
Tai150a	**3055.23**	3056.41	**3055.23**	0.0000	**3055.23**[a]
Tai150b	2727.67	2732.75	2755.09	3.7124	**2656.47**[c]
Tai150c	2343.11	2364.08	2352.86	0.4705	**2341.84**[a]
Tai150d	2645.40	2654.69	2660.33	0.5647	**2645.39**[a]
Tai385	24855.32	25015.01	25006.74	2.3547	**24431.44**[e]

[a] Taillard [5].
[b] Alba and Dorronosoro [26].
[c] Gambardella et al. [15].
[d] Mester and Bräysy [36].
[e] Rochat and Taillard [6]

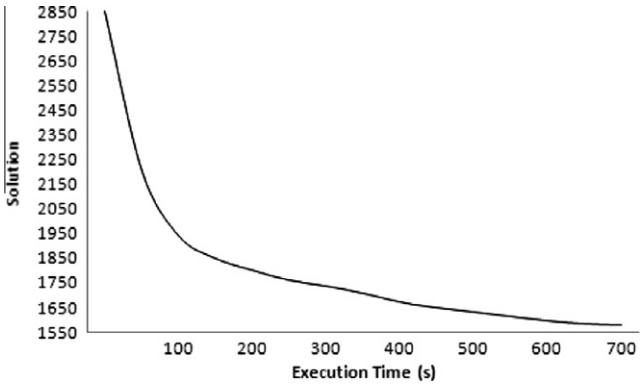

Fig. 4. Execution plot for instance Tai100d.

improved the best-so-far solution for one instance with 100 customers. The deviation value in this case, shows improvement of 0.0018% with respect to the best-so-far solution.

We plot the evolution of the best solution found by OCGA during a typical execution when solving instance Tai100d. As shown in Fig. 4, there is a fast convergence towards accurate solutions at the beginning of the execution, while in the rest of the search the evolution of the best solution is not that fast. The new best solution was found after 661 s in this run.

4. Conclusion

In this paper, we studied the capacitated vehicle routing problem, which is mainly characterized by using vehicles of the same capacity. We proposed a genetic algorithm using an optimised crossover operator for solving the CVRP. The computational experiments showed that the proposed algorithm is competitive in terms of the quality of the solutions found. As for future work, it may be interesting to test OCGA with other variants of VRPs.

References

[1] P. Toth, D. Vigo, The vehicle routing problem, in: P. Toth, D. Vigo, (Eds.), SIAM Monographs on Discrete Mathematics and Applications, Philadelphia, USA, 2002.
[2] G. Laporte, M. Gendreau, J.Y. Potvin, F. Semet, Classical and modern heuristics for the vehicle routing problem, International Transactions in Operations Research 7 (2000) 285300.
[3] J.F. Cordeau, M. Gendreau, G. Laporte, J.Y. Potvin, F. Semet, A guide to vehicle routing heuristics, Journal of the Operational Research Society 53 (2002) 512522.
[4] J.F. Cordeau, M. Gendreau, A. Hertz, G. Laporte, J.S. Sormany, New heuristics for the vehicle routing problem, in: A. Langevin, D. Riopel (Eds.), Logistics Systems: Design and Optimization, Springer, New York, 2005, p. 279297.
[5] É. Taillard, Parallel iterative search methods for vehicle routing problems, Networks 23 (8) (1993) 661–673.
[6] Y. Rochat, É. Taillard, Probabilistic diversification and intensification in local search for vehicle routing, Journal of Heuristics 1 (1995) 147–167.

[7] I.H. Osman, Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem, Operations Research 41 (1993) 421–451.
[8] M. Gendreau, A. Hertz, G. Laporte, A tabu search heuristic for the vehicle routing problem, Management Science 40 (1994) 1276–1290.
[9] C. Rego, C. Roucairol, A parallel tabu search algorithm using ejection chains for the vehicle routing problem, in: I. Osman, J. Kelly (Eds.), Meta-heuristics: Theory and Applications, Kluwer, Boston, 1996.
[10] G. Barbarosoglu, D. Ozgur, A tabu search algorithm for the vehicle routing problem, Computers & Operations Research 26 (1999) 255–270.
[11] P. Toth, D. Vigo, The granular tabu search and its application to the vehicle routing problem, INFORMS Journal on Computing. 15 (2003) 333348.
[12] D.T. Hiquebran, A.S. Alfa, A. Shapiro, D.H. Gittoes, A revised simulated annealing and cluster-first route-second algorithm applied to the vehicle routing problem, Engineering Optimization 22 (1994) 77–107.
[13] J. Renaud, F.F. Boctor, G. Laporte, An improved petal heuristic for the vehicle routing problem, Journal of the Operational Research Society 47 (1996) 329–336.
[14] B. Bullnheimer, R.F. Hartl, C. Strauss, An improved ant system algorithm for the vehicle routing problem, Annals of Operations Research 89 (1999) 319–328.
[15] L. Gambardella, É. Taillard, G. Agazzi, MACS-VRPTW: A multiple ant colony system for vehicle routing problems with time windows, in: D. Corne, M. Dorigo, F. Glover (Eds.), New Ideas in Optimization, McGraw-Hill, 1999, pp. 63–76.
[16] D. Mester, O. Bräysy, Active guided evolution strategies for large-scale capacitated vehicle routing problems, Computers & Operations Research 34 (2007) 2964–2975.
[17] C.D. Tarantilis, Solving the vehicle routing problem with adaptive memory programming methodology, Computers & Operations Research 32 (2005) 2309–2327.
[18] A.L. Chen, G.K. Yang, Z.M. Wu, Hybrid discrete particle swarm optimization algorithm for capacitated vehicle routing problem, Journal of Zhejiang University Science A 7 (2006) 607614.
[19] T.J. Ai, V. Kachitvichyanukul, A particle swarm optimization for the capacitated vehicle routing problem, International Journal of Logistics and SCM Systems. 2 (2007) 5055.
[20] T.J. Ai, V. Kachitvichyanukul, Particle swarm optimization and two solution representations for solving the capacitated vehicle routing problem, Computers & Industrial Engineering. 56 (2009) 380–387.
[21] J.Y. Potvin, D. Dubé, C. Robillard, A hybrid approach to vehicle routing using neural networks and genetic algorithms, Applied Intelligence 6 (1996) 241–252.
[22] J. Berger, M. Barkaoui, A hybrid genetic algorithm for the capacitated vehicle routing problem, in: E. Cant-Paz (Ed.), GECCO03, LNCS, vol. 2723, Springer-Verlag, Illinois, Chicago, USA, 2003, pp. 646–656.
[23] C. Prins, A simple and effective evolutionary algorithm for the vehicle routing problem, Computers & Operations Research 31 (2004) 1985–2002.
[24] B.M. Baker, M.A. Ayechew, A genetic algorithm for the vehicle routing problem, Computers & Operations Research 30 (2003) 787–800.
[25] E. Alba, B. Dorronosoro, Solving the vehicle routing problem by using cellular genetic algorithms, in: Evolutionary Computation in Combinatorial Optimization (EvoCOP), Lecture Notes in Computer Science, vol. 3004, Springer-Velag, Berlin, 2004, pp. 11–20.
[26] E. Alba, B. Dorronosoro, Computing nine new best-so-far solutions for capacitated VRP with a cellular genetic algorithm, Information Processing Letters 98 (2006) 225–230.
[27] 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 constrains. PhD Thesis. University of Antwerp, RCUA, Belgium, 1994.
[28] B. Golden, E. Wasil, J. Kelly, I.M. Chao, The Impact of Metaheuristics on Solving the Vehicle Routing Problem: Algorithms, Problem Sets, and Computational Results. Fleet Management and Logistics, Kluwer, Boston, 1998. pp. 33–56.
[29] J.H. Holland, Adaptations in Natural and Artificial Systems, The University of Michigan Press, Ann Arbor, 1975.
[30] C.C. Aggarwal, J.B. Orlin, R.P. Tai, Optimized crossover for the independent set problem, Operations Research 45 (1997) 226–234.
[31] E. Balas, W. Niehaus, Optimized crossover-based genetic algorithms for the maximum cardinality and maximum weight clique problems, Journal of Heuristics 4 (1998) 107–122.
[32] K.C. Tan, L.H. Lee, Q.L. Zhu, K. Ou, Heuristic method for vehicle routing problem with time windows, Artificial Intelligence in Engineering 15 (2001) 281–295.
[33] N. Christofides, A. Mingozzi, P. Toth, The vehicle routing problem, in: N. Christofides, A. Mingozzi, P. Toth, (Eds.), Combinatorial Optimization, 1979, pp. 315–338.
[34] <http://neo.lcc.uma.es/radi-aeb/WebVRP/Problem_Instances/CVRPInstances.html>.
[35] M. Reimann, K. Doerner, R. Hartl, D-ants: saving based ants divide and conquer the vehicle routing problem, Computers & Operations Research 31 (2004) 563–591.
[36] D. Mester, O. Bräysy, Active guided evolution strategies for large-scale vehicle routing problems with time windows, Computers & Operations Research 32 (2005) 1593–1614.