Available online at www.elixirpublishers.com (Elixir International Journal)

Applied Mathematics

Elixir Appl. Math. 71 (2014) 24911-24914

ISSN: 2229-712X

Vehicle routing problem with time windows using hybrid encoding genetic algorithm

S.Aruna Devi and P.K.Chenniappan

Department of Mathematics, Sri Shakthi Institute of Engineering and Technology, Tamil Nadu, India.

ARTICLE INFO

Article history:

Received: 6 February 2014;

Received in revised form:

6 June 2014;

Accepted: 12 June 2014;

Keywords

Vehicle routing Problem,

Time windows,

Hybrid Encoding Genetic Algorithm (HEGA),

Cross over and Mutation.

ABSTRACT

The vehicle routing problem is to determine K vehicle routes, where a route is a tour that begins at the depot, traverses a subset of the customers in a specified sequence and returns to the depot. Each customer must be assigned to exactly one of the K vehicle routes and total size of deliveries for customers assigned to each vehicle must not exceed the vehicle capacity. The routes should be chosen to minimize total travel cost. This paper gives a solution to find an optimum route for vehicle routing problem using Hybrid Encoding Genetic Algorithm (HEGA) technique. The objective is to find routes for the vehicles to service all the customers at a minimal cost without violating the capacity, travel time constraints and time window constraints

Introduction

Hybrid encoding genetic algorithm

As the standard GA are not suitable to implement Vehicle Routing Problem with Time Windows (VRPTW) in this study, new GA, called "Hybrid Encoding Genetic Algorithm (HEGA)" is developed for the problem of VRPTW.

The HEGA merges binary encoding that represent streets and integer encoding that represent customers. The creation of a new generation of chromosomes involves primarily four major steps: selection, crossover, mutation and reproduction. In the selection step, the roulette wheel selection was employed. In crossover step, only the one-point crossover for binary encoding was used. In mutation step, only the exchange mutation for integer encoding was performed and in reproduction step, the best chromosome is copied from the previous generation to the next one. The HEGA can be modified easily to handle VRPTW problem. The general structure of the HEGA can be described as follows:

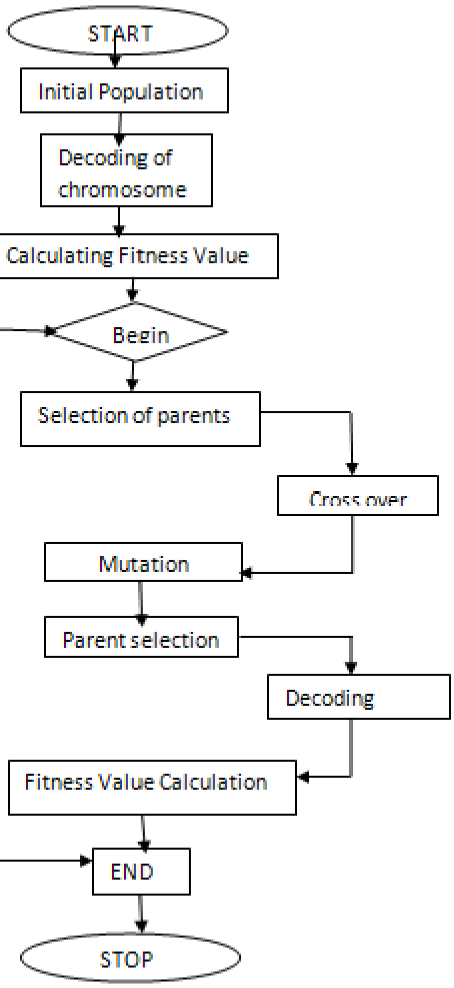


Fig 1.1. Flowchart

Vehicle Routing Problem

The Vehicle Routing Problem with Time Windows (VRPTW) which is an extension of Vehicle Routing Problems (VRPs) arises in a wide array of practical decision making problems. Instances of the VRPTW occur in rail distribution, airline distribution, school bus routing, mail and newspaper delivery and railway fleet routing and etc.

In general the VRPTW is defined as follows:

$V = \{1, 2, \ldots, K\}$, where V represents identical vehicles, a central depot node as D, a set of customer nodes $C = \{0,1,2,\ldots, N\}$ and a directed network connecting the depot and customers. Each arc in the network represents a connection between two nodes and also indicates the direction it travels. The depot is denoted as customer 0, which uses K independent delivery vehicles, with delivery capacity q_k, $k = 1,2,\ldots,K$, to service demands m_i from n customers, $i = 1,2,\ldots,N$.

The time window constraint is denoted by a predefined time interval $[a_i, b_i]$, a_i and b_i describe earliest arrival time and latest arrival time for customer i, respectively. The

Tele:

E-mail addresses: arunadevi0707@gmail.com

Vehicles must arrive at the customers not later than the latest arrival time, if vehicles arrive earlier than the earliest arrival time, waiting occurs. Each customer also imposes a service time s_i, taking consideration of the loading/unloading time of goods. A non-negative cost (distance or travel time) matrix $C = (c_{ij})$ between customers i and j is defined on the network. A solution for the VRPTW would be a partition $R_1, R_2, \ldots, R_K$, representing the routes of the vehicles, each route R_k is a permutation of the customers in C specifying the order of visiting them, starting and ending at the depot. The cost of the problem solution is the sum of the costs of its routes R_k, defined as follows $\sum_{k=1}^{k} \text{cost}(R_K)$

The VRPTW consists in determining a set of a maximum of K routes (i) of minimum total cost (Eq. 1); (ii) starting and ending at the depot denoted with customer 0 and such that (iii) each customer is visited exactly once by exactly one vehicle subject to the restrictions (iv) the total demand of any route R_k does not exceed q_k; (v) each route R_k must be completed within a total route time, which is essentially the time window of the depot; (vi) the vehicles must visit the customers within predefined time windows.

In this study, we consider the VRPTW and propose an HEGA for this problem.

Iii Decoding Using Hega

The representation of a solution we use here is a integer strength of length 'N', where 'N' is the number of customers. Each gene in the integer node number assigned to that customer originally. The sequence of the genes in the string is the order of visiting these customers.

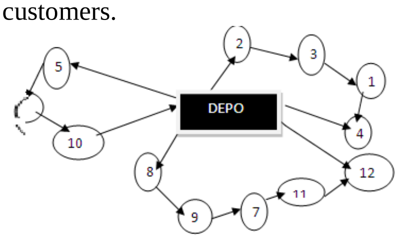


FIG (1.2) Network without overlapping

In the above Fig (1.1) the representation is unique and one string can be decoded to one-one solution. It is a one-one relation. Form number of sub networks without overlapping for the problem to get an optimum solution. One such sub network is considered for example. The last customer visited in route 'I' is linked with the depot. One string representing one cycle starts and ends in the depot. Each cycle have minimum of two possible routes say one in forward and in one in backward. To decode the string into route configurations the gene values are inserted in binary code format.

In the above fig (1.2) there are 12 customers forming 3 distinguished networks. 'D' represents Depo as '0'. Route 1 (R1) is represented as 0---2---3---1---4---0.Similarly Route 2 (R2) and Route 3 (R3) is represented as 0---5---6---10---0 and 0---8---9---7---11---12---0. (R1)+(R2)+(R3) forms the coded integer for one solution as

0—2—3—1—4—5—6—10—8—9—7—11—12—0

The above code can be binary encoded as follows: (R1)---000; (R2)---010;(R3)---100. Combining (R1), (R2) and (R3) binary encodes with Depot we get Chromosome1(Ch1) as defined in HEGA algorithm

D	0	0	0	R1	0	1	0	R2	1	0	0	R3	0	1	1	D

Ch1

which can be taken as

Ch1

D	0	0	0	1	0	1	0	2	1	0	0	3	0	1	1	D

Similarly Ch2, Ch3, Ch4 etc can be obtained by combining different possible solutions of R1, R2, R3 in all the possible ways (no overlapping between the network cycles).

Ch2

D	0	1	1	2	0	0	0	3	1	1	1	1	1	0	0	D

Ch3

D	0	1	0	3	1	1	1	1	1	0	1	2	1	0	1	D

Ch4

D	1	1	1	1	0	1	1	3	1	1	0	2	1	1	1	D

Fig (1.3)-Chromosome Decoding

Considering all possible combinations and randomly encode it only by using 3 bit binary digit.Sequence of network can be changed in all possible permutation ways (as R2,R3 &R1; R1,R3 &R2; R`3,R1&R2; etc) as each one is one chromosome. All chromosomes together form a initial population by sampling the matrix of the population.The integer number with underline means the vertices (no vertices at '0')and without underline as edges. The number of rows represent size of population. Considering the following example:

$$\begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 1 & 0 & 2 & 1 & 0 & 0 & 3 & 0 & 1 & 1 \\ 0 & 1 & 1 & 2 & 0 & 0 & 0 & 3 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 3 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 2 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 & 1 & 1 & 3 & 1 & 1 & 0 & 2 & 1 & 1 & 1 \end{bmatrix}$$

Decoding above from binary to decimal we have as

$$\begin{bmatrix} 4 & 1 & 2 & 2 & 4 & 3 & 3 \\ 3 & 2 & 0 & 3 & 7 & 1 & 4 \\ 2 & 3 & 7 & 1 & 5 & 2 & 5 \\ 7 & 1 & 3 & 3 & 6 & 2 & 7 \end{bmatrix}$$

Repeat decoding by comparing the first and the final vertex that has 3 different weights. The weights 0,1,2 is for first edge. For 3,4,5 we have second and for 6,7 we have third edges respectively . Then the decoded matrix becomes as

$$\begin{bmatrix} 2 & 1 & 1 & 2 & 2 & 3 & 2 \\ 2 & 2 & 0 & 3 & 3 & 1 & 2 \\ 1 & 3 & 3 & 1 & 3 & 2 & 3 \\ 3 & 1 & 2 & 3 & 3 & 2 & 3 \end{bmatrix}$$

Selection Of Parents From Population

The objective value is further calculated by using the objective function relating to each of the problem. In VRPTW low objective value is required for example the objective value matrix can be taken as

$$\begin{bmatrix} 49 \\ 35.6 \\ 32.2 \\ 26.2 \end{bmatrix}$$

From above the chromosome with lowest objective value is which is the best chromosome. The best chromosome with most fitness value (lowest objective value) survive for the next generation. For example,

Using Roulette wheel method of selection select the chromosomes from the population. Probably give highest fitness value to the chromosome that survive for the next generation and lowest fitness value to the one which is not be survived and remove it from the population. So only best fits will move to the next generation.

Cross-Over

From two selected parents cross-over is done and two off-springs are produced. As we have only binary codes to be crossed one point cross over is applied as defined in HEGA. Consider the following for example:

D	0	1	1	2	0	0	0	3	1	1	1	1	1	0	0	D

Parent1

D	0	1	0	3	1	1	1	1	1	0	1	2	1	0	1	D

Parent2

D	0	1	1	2	0	0	0	3	1	0	1	1	1	0	1	D

Offspring1

D	0	1	0	3	1	1	1	1	1	1	1	2	1	0	0	D

Offspring2

Fig (1.4) – Cross Over

Interchanging of binary bits alone between 3—1—D of parent1 and 1—2—D of Parent2 generates offspring1 and offspring2 as above.

Mutation

To perform mutation for the binary codes we used Exchange mutation as defined in HEGA.

D	1	1	1	1	0	1	1	3	1	1	0	2	1	1	1	D

Parent 1

D	1	1	1	2	0	1	1	3	1	1	0	1	1	1	1	D

Offspring

Fig (1.5)-Mutation

Reproduction

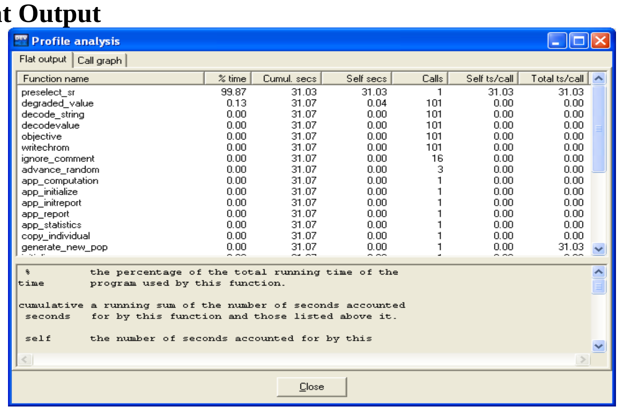
All offspring generated are sent to population and process repeated again copying the best chromosome under the following conditions: Travelling cost and time should be minimized on visiting each customer only once.

- Travelling time should not be greater than the latest arrival time
- The depot uses 'K' independent delivery vehicles with capacity q_k, q_k >k is not allowed.

Computational results:

We have implemented the above algorithm in C programming and got the following results

Flat Output


Profile analysis
Flat output | Call graph

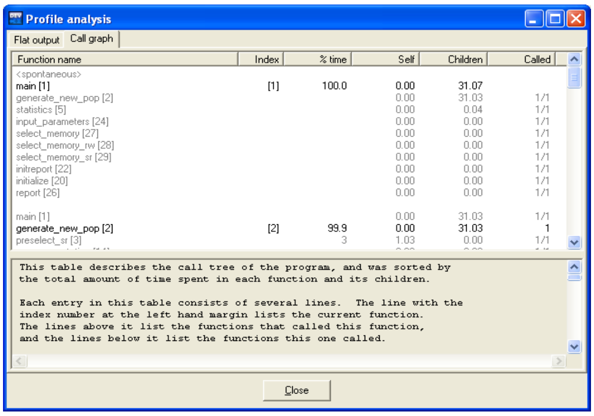
Function name	% time	Cumul. secs	Self secs	Calls	Self ts/call	Total ts/call
preselect_sr	99.87	31.03	31.03	1	31.03	31.03
degraded_value	0.13	31.07	0.04	101	0.00	0.00
decode_string	0.00	31.07	0.00	101	0.00	0.00
decodevalue	0.00	31.07	0.00	101	0.00	0.00
objective	0.00	31.07	0.00	101	0.00	0.00
writechrom	0.00	31.07	0.00	101	0.00	0.00
ignore_comment	0.00	31.07	0.00	16	0.00	0.00
advance_random	0.00	31.07	0.00	3	0.00	0.00
app_computation	0.00	31.07	0.00	1	0.00	0.00
app_initialize	0.00	31.07	0.00	1	0.00	0.00
app_initreport	0.00	31.07	0.00	1	0.00	0.00
app_report	0.00	31.07	0.00	1	0.00	0.00
app_statistics	0.00	31.07	0.00	1	0.00	0.00
copy_individual	0.00	31.07	0.00	1	0.00	0.00
generate_new_pop	0.00	31.07	0.00	1	0.00	31.03

```
 %         the percentage of the total running time of the
time       program used by this function.

cumulative a running sum of the number of seconds accounted
 seconds   for by this function and those listed above it.

 self      the number of seconds accounted for by this
```
Close

(Fig 1.6)


Profile analysis
Flat output | Call graph

Function name	Index	% time	Self	Children	Called
<spontaneous>					
main [1]	[1]	100.0	0.00	31.07	
generate_new_pop [2]			0.00	31.03	1/1
statistics [5]			0.00	0.04	1/1
input_parameters [24]			0.00	0.00	1/1
select_memory [27]			0.00	0.00	1/1
select_memory_rw [28]			0.00	0.00	1/1
select_memory_sr [29]			0.00	0.00	1/1
initreport [22]			0.00	0.00	1/1
initialize [20]			0.00	0.00	1/1
report [26]			0.00	0.00	1/1
main [1]			0.00	31.03	1/1
generate_new_pop [2]	[2]	99.9	0.00	31.03	1
preselect_sr [3]		3	1.03	0.00	1/1

```
This table describes the call tree of the program, and was sorted by
the total amount of time spent in each function and its children.

Each entry in this table consists of several lines.  The line with the
index number at the left hand margin lists the current function.
The lines above it list the functions that called this function,
and the lines below it list the functions this one called.
```
Close

Fig 1.7 Call Graph

Discussion

From the above output it is observed that the time required in calculating the shortest path is much lesser (milliseconds) by using HEGA. On comparing the result obtained with the previous results of the same problem by using various best algorithms such as "An improved hybrid genetic algorithm for the vehicle routing problem with time windows." (Berger, J. and M. Barkaoui, 2000.), "Vehicle routing problem with time windows, (Bräysy, O. and M. Gendreau, 2005.)," . "Optimized crossover genetic algorithm for vehicle routing problem with time windows, (Nazif, H. and L.S. Lee, 2010." The computational results shows minimum time in HEGA algorithm to find the shortest path.

We got the above result on taking a sample input of population size (10), Number of generations (5),and variables (10) with binary coded integer, fitness value, cross-over probability and mutation probability as input vales. For selection of population size the program uses Rolutte wheel selection method and One point cross over and exchange mutation are applied to get the best shortest path in minimum time. The sample size (Population) can also be increased to have maximum possible runs with more generations.

In future the size of the chromosomes taken (binary coded) can be increased to get still more best fit paths.

Conclusion

In this paper we have discussed the Vehicle Routing problem using Hybrid Encoding Genetic Algorithm. Various techniques of HEGA have been discussed in this paper to study Vehicle routing problem which is a permutation problem in which goal is to find the shortest path between customers visiting each customer at least once. This paper gives a solution to find an optimum route for VRPTW using HEGA technique, by selecting randomly the initial population. The new generations are then created repeatedly until the proper path is reached upon reaching the stopping criteria.

References

- Bent, R. and P. Van Hentenryck, 2004. A two-stage hybrid local search for the vehicle routing problem with time windows. Transport. Sci., 38: 515-530. DOI: 10.1287/trsc.1030.0049
- Berger, J. and M. Barkaoui, 2000. An improved hybrid genetic algorithm for the vehicle routing problem with time windows. Proceeding of the International ICSC Symposium on Computational Intelligence, Part of the International ICSC Congress on Intelligent Systems and Applications, Dec. 11-15, University of Wollongong, Wollongong, Australia.
- http://www.proceedings.com/00444.html

- Berger, J. and M. Barkaoui, 2004. A parallel hybrid genetic algorithm for the vehicle routing problem with time windows. Comput Operat. Res.,31: 2037-2053.DOI: 10.1016/S0305-0548(03)00163-199
- Berger, J., M. Salois and R. Begin, 1998. A hybrid genetic algorithm for the vehicle routing problem with time windows. Lecture Notes Artifi. Intell.,1418: 114-127. DOI: 10.1007/3-540-64575-6_44
- Berger, J., M. Sassi and M. Salois, 1999. A hybrid genetic algorithm for the vehicle routing problem with time windows and itinerary constraints. Proceedings of the Conference on Genetic and Evolutionary Computation, July 13-17, Orlando, USA., pp: 44-51. http://www.alibris.com/search/books/isbn/1558606114
- Blanton, J.L. and R.L. Wainwright, 1993. Multiple vehicle routing with time and capacity constraints using genetic algorithms. Proceedings of the 5th International Conference on Genetic Algorithms, (ICGA'93), Morgan Kaufmann, San Francisco, pp: 452-459.
- Bräysy, O. and M. Gendreau, 2005. Vehicle routing problem with time windows, Part II: Metaheuristics. Transport. Sci., 39: 119-139. DOI: 10.1287/trsc.1030.0057
- Bräysy, O., J. Berger and M. Barkaoui, 2000. A new hybrid evolutionary algorithm for the vehicle routing problem with time windows. Proceedings of the International Workshop on Vehicle Routing and Related Areas, Aug. 16-19, Skodsborg, Denmark. http://74.125.155.132/ scholar?q=cache:05I0XxcO08UJ:scholar.google.com/+Berger,+ J.+and+M.+Bar kaoui,+2000&hl=en&as_sdt=2000
- Chiang, W.C. and R.A. Russell, 1997. A reactive tabu search metaheuristic for the vehicle routing problem with time windows. Inform. J. Comput., 9: 417-430. DOI:10.1287/ ijoc.9.4.417
- Gendreau, M., G. Laporte and D. Vigo, 1999.Heuristics for the traveling salesman problem with pickup and delivery. Comput. Operat. Res., 26:699-714. DOI: 10.1016/S0305-0548(98)00085-9
- Kalantari, B., A.V. Hill and S.R. Arora, 1985. An algorithm for the traveling salesman problem with pickup and delivery customers. European J. Operat. Res., 22:377-386.DOI:0.1016/ 0377-2217(85)90257-7
- Nazif, H. and L.S. Lee, 2010. Optimized crossover genetic algorithm for vehicle routing problem with time windows. Am. J. Applied Sci., 7: 95-101. DOI: 10.3844/ajassp.2010.95.101
- Patvichaichod,S., 2011 An improved genetic algorithm for the traveling salesman problem with multi-relations. J. Comput. Sci., 7: 70-74. DOI: 10.3844/jcssp.2011.70.74
- I.M. Oliver, D. J. Smith and J.R.C. Holland. "A Study of Permutation Crossover Operators on the Travelling Salesman Problem". In J.J. Grefenstette (ed.). Genetic Algorithms and Their Applications: Proceedings of the 2nd International Conference on Genetic Algorithms. Lawrence Erlbaum Associates, Hilladale, NJ, 1987.