Proceedings of the 2007 IEEE Symposium on Computational Intelligence in Scheduling (CI-Sched 2007)

Improving the Performance of Genetic Algorithm in Capacitated Vehicle Routing Problem using Self Imposed Constraints

Ziauddin Ursani, Ruhul Sarker, and Hussein A. Abbass,

School of Information Technology and Electrical Engineering,
University of New South Wales,
ADFA Campus, Northcott Drive, Canberra, ACT 2600
Australia

***Abstract*— The Capacitated Vehicle Routing Problem (CVRP) is a well known member of the family of NP hard problems. In the past few decades, a number of heuristics was introduced to solve this problem but no heuristic can claim to work well in all possible scenarios. In the literature, Genetic Algorithm (GA) even lags behind the other heuristics. In this paper, we reveal some of the reasons for the inferior performance of GA, and propose a number of mechanisms to improve its performance. A number of test problems are solved to demonstrate the usefulness of the algorithm.**

I. Introduction

The Vehicle Routing Problem (VRP) was first introduced by Dantzig and Ramser [1], where the objective is to find the route with the lowest cost for a number of homogeneous vehicles, stationed at a depot, destined to satisfy heterogeneous demand of customers, situated at geographically dispersed locations, such that, the route of each vehicle should start and end at the depot, and each customer must be visited only once by only one vehicle. The capacitated version of the problem is known as CVRP which has the additional constraint that the route of any vehicle should not contain the set of customers with demand greater than the capacity of vehicle.

This problem has attracted enormous interest in the research community, partially due to the simple definition, practical importance and computational challenges to achieve optimality. A number of heuristics and meta-heuristics are applied to this problem. Those heuristics include Local and Neighborhood Search, Simple and Granular Tabu Search, Simulated and Deterministic Annealing, Ant colony and artificial life systems, evolutionary and population search etc. Some of the most successful work produced among these are Parallel and iterative search [2], Granular Tabu Search [3], Large Scale Neighborhood Search [4], [5] and Evolutionary Strategy [6]. People have also applied GAs and hybrid GAs [7]-[9]. Although GA is a very popular global optimization technique, it is not as successful as other heuristics on multi-route problems, particularly in the case of CVRP. In this paper we have analyzed the performance of GA and identified the reasons for its inferior performance in CVRP. We have also suggested a number of modifications to improve the performance of GA on CVRP. The performance of the proposed algorithm is validated by solving a number of well known bench mark problems that have appeared in the literature.

This paper is organized as follows: We analyze the problem in section II. In section III we have explained our algorithm. Experiments and results are detailed in section IV. Finally in section V conclusions and future work is specified.

II. Problem Analysis

GA performs relatively well on the traveling salesman problem (TSP) but not so well on VRP. Among many other reasons, ambiguous data representation is a key reason behind its inferior performance. To date, Path Representation (PR) is considered the most suitable representation. However it has proved to be suitable for single route problems only such as TSP. It is unable to perform well in multiple routes mainly because of its difficulty in identifying terminal points of each sub-route. We explain this point using an example. Let us consider that there are 9 customers numbered from 1 to 9, such that the demand of each customer is equivalent to its number ID. Now let us consider 235897614 is our chromosome. Let us also consider that each vehicle can accommodate a maximum demand of 25. Considering the above

The first author acknowledges financial support by a University College Post Graduate Research Scholarship, UNSW@ADFA.

Mr. Z. Ursani is a Ph.D. research candidate with the School of Information Technology and Electrical Engineering (ITEE), Australian Defence Force Academy (ADFA), University of New South Wales (UNSW). (phone: +61262688180; fax: +6162688581; e-mail: z.ursani@adfa.edu.au).

Dr. R. Sarker is a Senior Lecturer with the School of Information Technology and Electrical Engineering (ITEE), Australian Defence Force Academy (ADFA), University of New South Wales (UNSW). (phone: +61262688051; fax: +61262688581; e-mail: r.sarker@adfa.edu.au).

Dr. H. Abbass is an Associate Professor with the School of Information Technology and Electrical Engineering (ITEE), Australian Defence Force Academy (ADFA), University of New South Wales (UNSW). (phone: +61262688158; fax: +61262688581; e-mail: h.abbass@adfa.edu.au).

chromosome, the first route will consist of customers 2358, because if we add customer 9 to the route, the total demand of the route will become 27, exceeding the vehicle capacity of 25. The next route will consist of customers 9761. The last customer number 4 cannot be included in this route because if it does, the capacity constraint will be violated. Hence the last route will consist of customer 4 alone. By looking carefully at the chromosome, we can see that we can divide this chromosome in several different sub-routes while satisfying the capacity constraint and respecting the sequence altogether. For example, it can be divided into sub-routes 2358-976-14. These routes lie within the capacity limit of the vehicle. It can also be divided into legal routes 235-897-614, 235-89-7614, 23-589-7614, and 2-3589-7614. There are many other possibilities if we do not stick ourselves to 3-route limit. Therefore, how can we find which set of sub-routes represents the minimum cost solution? No clear answer found to this question in the literature. This is the major difficulty of representational issues in Genetic Algorithms, because of which it cannot compete with other heuristics. Prins [9] has come up with the idea of trying to test all possible subsets of routes to find out the best set in each evaluation. But this scheme is computationally expensive, with a complexity of order (*nb*), where *n* is the number of customers and *b* is the highest number of customers present in any route. If we consider the 5th test problem of Christofieds [10], which has 199 customers and 17 routes, then for one individual in a given generation, this scheme will require around 199x199/17 = 2330 normal fitness evaluations in its best case scenario. When compared to real world situations, this data set is relatively small; and the idea of Prins may fail in wider goods distribution network.

To deal with this problem, we have come up with a novel idea of imposing various constraints to identify the terminator for each sub-route (i.e. it tells the GA where to terminate the route). The terminator will be placed once any of these constraints is about to be violated. We have identified a number of constraints for this purpose, as discussed bellow:

- **Route Constraint (RC):** No route length should exceed particular limit.
- **Active Route Constraint (ARC):** Active route is the route length starting from first customer and ending at the last customer (omitting the leg distances from and towards the depot). We can set a particular limit for the active route also.
- **Edge Constraint (EC):** The Euclidean distance between any two consecutive customers should not exceed a particular limit.
- **Customer Number Constraint (CNC):** This constraint limits number of customers in any route.

After identifying the constraints the next step is to decide on suitable values for their parameters, which can guide the GA towards good solutions. It is obvious that values for these parameters will be different for each and every type of problem. Although it is difficult to set the values, we have successfully set them adaptively through a clustering scheme. Clustering can also be used in routing for initialization, but the proposed scheme has never been used in any routing problem from this point of view. The scheme is described below.

A. *Clustering Scheme*

This scheme is similar to the k-nearest neighbor algorithm with the addition of a capacity on each bin of the cluster. It consists of the following steps,

1) Randomly generate *n* points in the customer landscape, where *n* should be the upper integer of the ratio of the total customer demand to the vehicle capacity.

2) Allocate the customers to these points, in the order of first to last customer. The customer must be allocated to the point which is the nearest to it as compared to other points. These allocations will form *n* clusters. A customer cannot be allocated to a cluster if the total demand of the cluster exceeds the capacity constraint of a vehicle.

3) Calculate the centroid of each of the cluster.

4) Reshuffle the order of customers, through random insertion.

5) Taking these centroids as new points, repeat the steps 2 to 4, until the clustering stabilizes, i.e. difference between new and old centroids become close to zero.

This is also called crisp clustering as each customer is decisively assigned to a particular cluster [11]. The clustering scheme described above is suitable for standard CVRP only, i.e. the problem without externally imposed route constraint. It is so because we have considered only capacity constraint during clustering. But if we are solving advanced version of CVRP - i.e. problems with externally imposed route constraint - we will have also to consider route constraints along with the capacity constraint in point 2 of the above clustering scheme. To impose a route constraint we must estimate the optimized route length of the route concerned. To know this value, we can test the effect of adding a customer to each cluster before deciding on which cluster to add the customer to. This may increase the computational cost of clustering but will eliminate the need for optimization of individual routes after clustering.

B. *Self Imposed Constraints*

After having the optimized cluster paths, we can say that we have obtained the solution structure, which is not necessarily optimal but a better individual in a good shape, having some locally optimized properties. If we are able to identify those properties or parameters, these parameter values can be used in the GA as self imposed constraints for deciding on when to terminate a route when reading and evaluating a chromosome. To understand this point, consider 1st christofied's data set with 50 customers. After applying the above clustering scheme, we got the solution shown in figure-1. The best known solution found yet of the above data set is shown in figure-2.

By comparing the two solutions we can see that they are very similar in shape. Now let us compare the various properties of the two solutions, which are consolidated in table I.

TABLE I
PROPERTY COMPARISON BETWEEN TWO SOLUTIONS

Property	Solution after clustering	Best Solution
Longest Route	124.11	118.52
Longest Active Route	100.22	88.95
Longest Edge	20.10	15.03

As per table-1, the solution after clustering has slightly greater values of longest route, longest active route and longest edge as compared to the best known solution. If we use these values of the sample solution as self imposed constraints in our algorithm, then these constraints will terminate the sub-routes at more suitable places, where we can have low cost solutions. Consider the same previous example of section II, where we were dealing with the chromosome 235897614. Let us consider these customers form 3 different clusters i.e. 235, 897 and 614. Then the most suitable division of the chromosome into sub-routes will be 235-897-614. However if we consider only the capacity constraint, we will come up with a solution 2358-9761-4. Now if we apply self imposed constraints of longest route, longest active route and longest edge, the first route 2358 may violate one of these constraints and may end up to be 235. Naturally then, the next customer, 8, will join the second route. Again the second route will serve only the following 3 customers i.e. 897 due to its capacity constraint and the rest will move to the last route ending up in a desirable solution of 235-897-614.

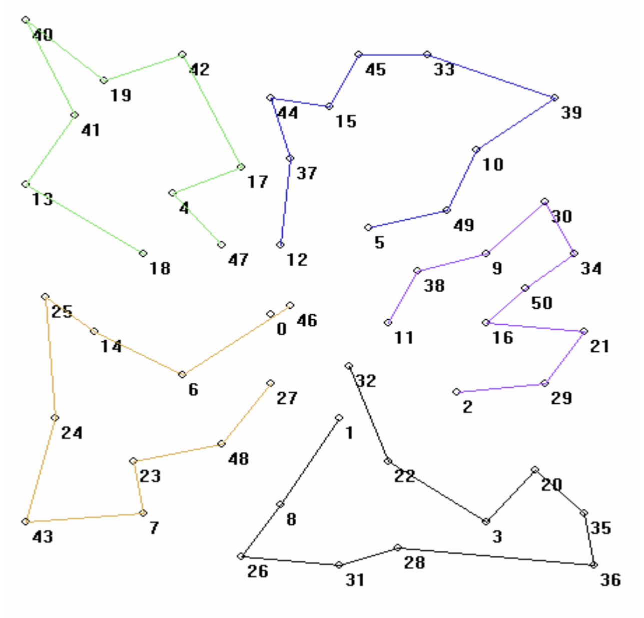

Fig. 1. Solution obtained after Clustering - Cost = 535.80

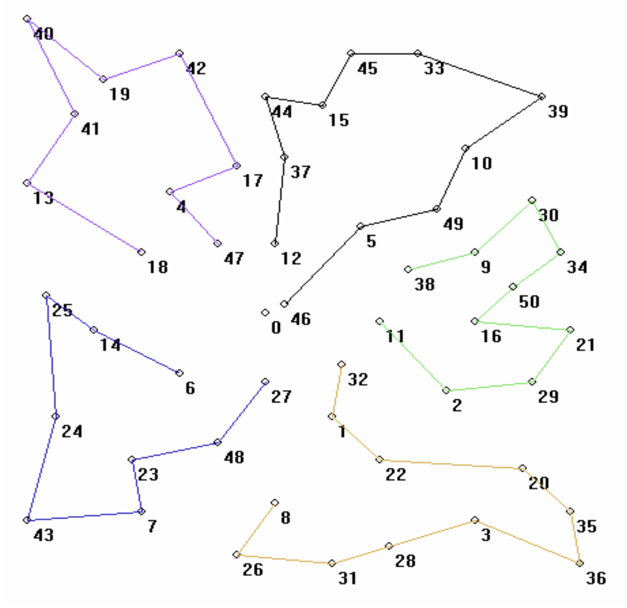

Fig. 2. Best Known Solution – Cost = 524.61

III. PROPOSED ALGORITHM

The algorithm consists of 3 phases, as shown in figure-3. The first phase is clustering, which is used to create initial population. It is followed by a local GA. We call it local GA (LGA) because it is applied to parts of the chromosome. The 3rd phase consists of local search (LS). If the solution is improved in this phase, then it goes back to the local GA and the procedure is repeated between LGA and LS until no improvement is possible in LS. All of these phases are described below in details.

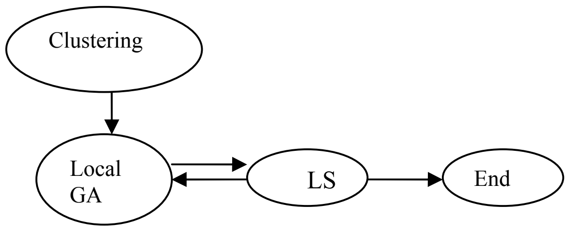

Fig. 3. Over all Algorithm Scheme

A. *Initial Population*

The initial population is created through the clustering scheme described in section II A. The population size is kept equal to the total number of customers. After the creation of the initial population, constraint values are calculated as described in section II B from the best individual of the population. The best individual is the one which gives the minimum sum of all individual route distances.

B. *Local Genetic Algorithm*

The local genetic algorithm is applied to all possible pairs of routes. We extract any two routes from the parent chromosome and form a mini-dataset. Then we apply LGA on that mini-dataset in two stages. In the first stage, all constraints are relaxed and in the second stage constraints are imposed. The first stage starts with random generation of initial population. The population size is made equal to the number of customers present in the mini-dataset. Tournament Selection of team size 2 (binary tournament) is used. The fitness cost is the total distance covered by vehicles, only one vehicle at this stage as all constraints are relaxed. Minimizing the fitness cost is our objective. For binary genetic operators our choice is partially mapped crossover. It is applied with probability equal to 0.6. Combination of 3 unary operators, inversion, insertion and swap mutations are applied with probabilities equal to 0.15, 0.1 and 0.05 respectively. Cloning of individuals is done with probability of 0.1. The termination condition is set as the number of non-improved generations equal to 5 times the population size. Second stage gets evolved population from the first. All the other parameters remain the same in the second stage. But since all the constraints are applied at this stage, the fitness cost will be the distances covered by one or more vehicles whatever the case may be. Note that the local GA is applied on the best solution of the population only. If local GA brings multiple changes to the solution, then multiple copies of solutions are developed each with a single change. The best solution from these individuals is selected for the application of LGA again. The process is repeated till no improvement is possible through LGA.

C. *Local Search*

Local Search is applied to all the individuals, which are products of local GA. It consists of the following operations.

1. Swap: Position of two or more consecutive customers is swapped. In the usual neighborhood search terminology it is called λ-opt neighborhood, if customers belong to the same route and λ-interchange neighborhood if customers belong to different routes [5], where λ is the number of customers swapped.

2. Insertion: One or more customers are taken out from their position and inserted in another position. In terms of neighborhood search, it is called string relocation [5].

3. Inversion: Whole segment of customers between two positions is inverted.

4. Swap-unequal: Two unequal strings of customers are swapped. It is called string exchange in neighborhood search [5].

The following new operations are introduced to enlarge the neighborhood.

5. Swap-Partially Inverted (SPI): It is a combination of segment exchange and inversion. A position of two equal or unequal strings of customers are swapped, afterwards one of them is inverted.

6. Swap-Fully Inverted (SFI): It is the same operation as 5, except that, both the swapped strings are inverted this time.

7. Insertion-Inverted (II): It is a combination of inversion and insertion. A string of customers is taken out from their position and inserted at another place. Then it is inverted.

IV. Experiments and Results

We coded the above algorithm in C++, with Microsoft visual C++ v6 compiler. The operating system was Microsoft Windows XP professional, version 2002. The program was executed on Pentium-4, 3.2 GHz processor with 1 GB RAM.

In our experiments, to see the effect of each and every phase of our algorithm, we applied our algorithm in parts and have studied their effect over the best and average results. First we applied only phase-1 i.e. clustering. Table II shows the results of these experiments. In our next set of experiments, Local GA was applied after clustering. The results of these experiments are shown in Table III. In our next set of experiments, Local Search was also introduced to improve the solution. The algorithm was allowed to iterate between local GA and Local search as shown in figure-3. The results of such experiments are shown in Table IV. In our final set of experiments we allowed 1% deterioration of solution through our local search mechanism. We also introduced another self imposed constraint, i.e., Customer Number Constraint. The results of such experiments are shown in Table V.

For the above experiments, we chose 14 data sets of Christofied. These results are based on the best solutions among 30 program executions, which were run on different randomly generated seeds. In these tables, column 1 shows the serial number of the data set, column 2 shows the number of customers, present in the data set. Column 3 shows the value of the best known solution appeared in the literature. Column 4a shows the minimum solution value in 30 executions of the program, whereas column 4b shows the difference in percentage of this solution against the best known solution in literature. Finally columns 5a, 5b and 5c show the average solution values of all 30 results, their difference in percentage from best known solutions and the average cpu times in seconds respectively.

One can see that after the clustering, the average of best values obtained is 8.61% away from the best known solutions. After the application of local GA, this value decreases to 1.02%. The value is reduced to 0.85% after the application of Local GA – Local Search cycle and finally it touches the value of only 0.45% after allowing 1% solution deterioration and introduction of an additional self imposed constraint, i.e. a customer number constraint. This proves that each part of algorithm contributes to the solution quality positively and has its important role to play. It should be noted that self imposed constraints are at work in both the sections i.e. Local GA and Local Search.

TABLE II
RESULTS AFTER CLUSTERING

1	2 .	3	4		5		
			Minimum		Average		
	Nr. of	Best	a	b	A	b	c
Sr. Nr.	Custs	Known	Dist	%diff	Dist	%diff	Time
1	50	524.61	524.61	0.00	533.44	1.68	0.42
2	75	835.26	881.32	5.51	903.18	8.13	10.82
3	100	826.14	884.65	7.08	892.24	8.00	8.44
4	150	1028.42	1132.7	10.14	1141.6	11.00	50.03
5	199	1291.29	1462.7	13.28	1497.1	15.94	310.22
6	50	555.43	567.32	2.14	578.25	4.11	0.81
7	75	909.68	957.56	5.26	976.57	7.35	7.39
8	100	865.94	950.31	9.74	977.99	12.94	14.63
9	150	1162.55	1300.5	11.87	1331	14.49	88.02
10	199	1395.85	1592.0	14.05	1643	17.71	278.08
11	120	1042.11	1068.8	2.56	1082.4	3.86	20.83
12	100	819.56	829.32	1.19	830.61	1.35	0.44
13	120	1541.14	2061.8	33.79	2167.5	40.65	21.5
14	100	866.37	900.39	3.93	916.02	5.73	16.61
Avg				8.61		10.93	59.16

TABLE III
RESULTS AFTER CLUSTERING AND LOCAL GA

1	2	3	4		5		
			Minimum		Average		
	Nr. of	Best	a	b	a	b	c
Sr. Nr.	Custs	known	Dist	%diff	Dist	%diff	Time
1	50	524.61	524.61	0.00	526.74	0.41	1.92
2	75	835.26	845.46	1.22	860.64	3.04	18.01
3	100	826.14	840.76	1.77	849.30	2.80	32.40
4	150	1028.42	1059.40	3.01	1069.93	4.04	141.61
5	199	1291.29	1318.01	2.07	1335.87	3.45	579.98
6	50	555.43	555.43	0.00	564.49	1.63	2.81
7	75	909.68	930.64	2.30	945.32	3.92	12.78
8	100	865.94	867.41	0.17	892.14	3.03	42.98
9	150	1162.55	1182.33	1.70	1204.65	3.62	194.16
10	199	1395.85	1416.64	1.49	1453.05	4.10	547.08
11	120	1042.11	1043.57	0.14	1049.64	0.72	70.26
12	100	819.56	819.56	0.00	821.29	0.21	10.86
13	120	1541.14	1546.14	0.32	1571.62	1.98	121.39
14	100	866.37	866.79	0.05	875.97	1.11	36.96
Avg				1.02		2.43	129.51

Now let us compare our results with 3 popular GAs [7]-[9] applied to VRP, as appeared in the literature. The comparisons are shown in Table-VI. The columns only show percentage gap from best known solutions found so far. Percentages of the 5th data set are recalculated for all algorithms because a new solution [6] was found after the publication of these algorithms. In the paper of Baker & Ayechew [7], there was miscalculation in percentage differences in solutions of 5th and 10th data sets, it is also corrected here. Values from Christian Prins paper [9] are given which he obtained under one setting, because we have not configured settings of our algorithm separately for different data sets. From the table, one can easily see that the algorithm of self imposed constraints have got better solutions than two GAs [7], [8]. However solutions still leg behind the 3rd GA [9]. But as we discussed earlier in our problem analysis, that evaluation process of Prins algorithm i.e. checking all possible subsets of routes is computationally expensive and may fail in real world scenario, where we are dealing with several thousand customers with several hundred routes, therefore our algorithm can still be considered competitive and promising.

TABLE IV
RESULTS AFTER CLUSTERING AND LOCAL GA-LOCAL SEARCH CYCLE

1	2	3	4		5		
			Minimum		Average		
	Nr. of	Best	a	b	a	b	c
Sr. Nr.	Cust.	known	Dist	%diff	Dist	%diff	Time
1	50	524.61	524.61	0.00	526.71	0.40	1.83
2	75	835.26	845.46	1.22	859.76	2.93	19.69
3	100	826.14	839.60	1.63	846.71	2.49	41.89
4	150	1028.42	1042.78	1.40	1065.41	3.60	216.40
5	199	1291.29	1313.53	1.72	1330.72	3.75	788.05
6	50	555.43	555.43	0.00	563.73	1.49	2.81
7	75	909.68	930.64	2.30	944.02	3.78	17.22
8	100	865.94	867.41	0.17	888.95	2.66	60.17
9	150	1162.55	1182.12	1.68	1201.60	3.36	259.63
10	199	1395.85	1416.64	1.49	1446.95	3.66	747.19
11	120	1042.11	1042.11	0.00	1046.72	0.44	119.00
12	100	819.56	819.56	0.00	821.29	0.21	15.26
13	120	1541.14	1546.14	0.32	1569.94	1.87	165.00
14	100	866.37	866.37	0.00	875.13	1.01	47.85
Avg				0.85		2.26	178.72

TABLE V
RESULTS AFTER ALLOWING 1% DETERIORATION OF SOLUTION

1	2	3	4		5		
			Minimum		Average		
	Nr. of	Best	a	b	a	b	c
Sr. Nr.	Cust.	known	Dist	%diff	Dist	%diff	Time
1	50	524.61	524.61	0.00	524.84	0.04	3.20
2	75	835.26	835.77	0.06	855.53	2.43	32.00
3	100	826.14	833.14	0.85	841.09	1.81	85.13
4	150	1028.42	1038.65	0.99	1057.30	2.81	344.20
5	199	1291.29	1315.72	1.89	1333.00	3.23	1111.37
6	50	555.43	555.43	0.00	557.59	0.92	4.40
7	75	909.68	912.89	0.35	926.26	1.82	25.20
8	100	865.94	866.87	0.11	880.44	1.67	86.13
9	150	1162.55	1170.67	0.70	1197.08	2.97	395.10
10	199	1395.85	1410.66	1.06	1437.90	3.01	1382.10
11	120	1042.11	1042.11	0.00	1046.76	0.45	163.43
12	100	819.56	819.56	0.00	820.01	0.06	26.30
13	120	1541.14	1545.4	0.28	1565.98	1.61	209.23
14	100	866.37	866.37	0.00	868.67	0.27	61.30
Avg				0.45		1.65	280.65

TABLE VI
COMPARATIVE RESULTS WITH OTHER GAS

1	2	3	4	5	6
Sr. Nr.	Nr. of Custs	Baker & Ayechew	Berger & Barkaoui	Christian Prins	Self Imposed Constraints
1	50	0.00	0.00	0.00	0.00
2	75	0.43	0.00	0.00	0.06
3	100	0.40	0.15	0.00	0.85
4	150	0.62	0.75	0.31	0.99
5	199	2.83	2.54	0.69	1.89
6	50	0.00	0.00	0.00	0.00
7	75	0.00	0.00	0.29	0.35
8	100	0.20	0.27	0.00	0.11
9	150	0.32	0.57	0.15	0.70
10	199	2.11	1.64	1.70	1.06
11	120	0.46	0.10	0.00	0.00
12	100	0.00	0.00	0.00	0.00
13	120	0.34	0.78	0.12	0.28
14	100	0.09	0.00	0.00	0.00
Avg		0.56	0.49	0.23	0.45

V. CONCLUSION AND FUTURE WORK

By looking at the results, one can easily conclude that the algorithm scheme is quite suitable for the vehicle routing problem. This research was intended to identify problems with GA while dealing with multiple routes. We identified some problems, such as difficulties in the identification of terminal genes of sub-routes and proposed some new mechanisms like self imposed constraints to deal with those difficulties. However the paradigm of self imposed constraints in the vehicle routing problem is still nascent and needs to be developed further. We will continue to investigate possibilities for other self imposed constraints, such as the largest angle subtended by any edge, with respect to the depot. If these new constraints are designed and applied carefully, we may be able to improve the results further. There also remains the great potential for reducing the computational cost of the algorithm. For this, one may consider the potential of using the clustering technique to estimate the targets for self-imposed constraints without using clustering for initialization. The idea of self imposed constraints should also be applied on other variants of VRP such as VRP with time windows etc, to test its robustness. The Self Imposed Constraints also need to be supported by some theoretical work to expand its area of application.

REFERENCES

[1] G. B Dantzig,, J. H. Ramser, “The truck dispatching problem,” Management Science Vol. 6, 1959, 80.
[2] E. Taillard, “Parallel Iterative Search Methods for Vehicle Routing Problems,”. Networks, Vol. 23, 1993, pp. 661-673.
[3] P . Toth, D. Vigo, “The Granular Tabu Search and Its Application to the Vehicle-Routing Problem,” INFORMS Journal on Computing, Vol. 15, 2003, pp. 333-346.
[4] F. Li, B. Golden., E. Wasil, “Very Large Scale Vehicle Routing: new test problems, algorithms and results,” Computers and Operations Research Vol. 32, 2005, pp. 1165-1179.
[5] R. Agarwal, R. K Ahuja., G. Laporte, Z. J. M. Shen, “A composite Very Large Scale Neighborhood Search Algorithm for the Vehicle Routing Problem,” Handbook of Scheduling, Algorithms, Models, and Performance Analysis. Chapter 49, 2004.
[6] D. Mester, O. Braysy, “Active Guided Evolutionary Strategy for the Large Scale Vehicle Routing Problems with Time Windows,” Computers and Operations Research, Vol. 32, 2005, pp. 1593-1614.
[7] B. M. Baker, M.A. Ayechew, “A genetic algorithm for the vehicle routing problem,” Computers & Operations Research, Vol. 30, 2003, pp. 787-800.
[8] J. Berger, M. Barkaoui, “A Hybrid Genetic Algorithm for the Capacitated Vehicle Routing Problem,” GECCO LNCS, 2003, pp. 646-656.
[9] C. Prins, “A simple and effective evolutionary algorithm for the vehicle routing problem,” Computers & Operations Research, Vol. 31, 2004, pp. 1985-2002.
[10] N. Christofides, A. Mingozzi,, P. Toth,, “The vehicle routing problem,” In: N. Christofides, A. Mingozzi,, P. Toth, , C Sandi. (eds.): Combinatorial Optimisation. Wiley, chichesster, 1979.
[11] H. Maria, B. Yannis, V. Michalis, “On Clustering Validation Techniques” Journal of Intelligent Information Systems, Vol. 17, 2001, pp. 107-145.
[12] Z. Michalewicz,, “Genetic Algorithms + Data Structures = Evolution Programs,” 3rd edn. Springer-Verlag, Berlin Heidelberg New York, 1996.