Solving the Vehicle Routing Problem using Genetic Algorithm

Abdul Kadar Muhammad Masum[1]
Dept. of Business Administration
International Islamic University Chittagong

Mohammad Shahjalal[2]
Dept. of Electrical & Electronic Engineering
International Islamic University Chittagong

Md. Faisal Faruque[3]
Dept. of Computer Science & Engineering
University of Information Technology & Sciences

Md. Iqbal Hasan Sarker[4]
Dept. of Computer Science & Engineering
Chittagong University of Engineering & Technology

Abstract— **The main goal of this research is to find a solution of Vehicle Routing Problem using genetic algorithms. The Vehicle Routing Problem (VRP) is a complex combinatorial optimization problem that belongs to the NP-complete class. Due to the nature of the problem it is not possible to use exact methods for large instances of the VRP. Genetic algorithms provide a search technique used in computing to find true or approximate solution to optimization and search problems. However we used some heuristic in addition during crossover or mutation for tuning the system to obtain better result.**

Keywords: Vehicle Routing Problem (VRP); Genetic Algorithm; NP-complete; Heuristic.

I. INTRODUCTION

The VRP can be described as follows: given a fleet of vehicles with uniform capacity, a common depot, and several customer demands, finds the set of routes with overall minimum route cost which service all the demands [1]. All the itineraries start and end at the depot and they must be designed in such a way that each customer is served only once and just by one vehicle. Genetic algorithms have been inspired by the natural selection mechanism introduced by Darwin [2]. They apply certain operators to a population os solutions of the problem at hand, in such a way that the new population is improved compared with the previous one according to a pre-specified criterion function. This procedure is applied for a pre-selected number of iterations and the output of the algorithm is the best solution found in the last population or, in some cases, the best solution found during the evolution of the algorithm. In general, the solutions of the problem at hand are coded and the operators are applied to the coded versions of the solutions. The way the solutions are coded plays an important role in the performance of a genetic algorithm. Inappropriate coding may lead to poor performance. The operators used by genetic algorithms simulate the way natural selection is carried out. The most well-known operators used are the reproduction, crossover, and mutation operators applied in that order to the current population. The reproduction operator ensure that, in probability, the better a solution in the current population is, the more (less) replicates it has in the next population. The crossover operator, which is applied to the temporary population produced after the application of the reproduction operator, selects pairs of solutions randomly, splits them at a random position, and exchanges their second parts. Finally, the mutation operator, which is applied after the application of the reproduction and crossover operators, selects randomly an element of a solution and alters it with some probability. Hence genetic algorithms provide a search technique used in computing to find true or approximate solutions to optimization and search problems.

II. SOLUTION DETAILS

At the beginning an initial generation has to be defined. This can be done using a random initialization or can use some kind of seeding which allows the algorithm to work in a search space where solutions are more likely. From now until a valid solution is found or the maximal level of allowed generations is reached, the following steps are performed.

Selection: first we select a proportion of the existing population to breed a new generation. The selection is done on a fitness-based approach where fitter individuals are more likely to breed then others.

Reproduction: during the reproduction phase the next generation is created using the two basic methods, crossover and mutation. For every new child a pair of parents is selected from which the child inherits its properties. In the crossover process genotype is taken from both parents and combined to create a new child.

With a certain probability the child is further exposed to some mutation, which consists of modifying certain genes. This helps to further explore the solution space and ensure, or preserve, genetic diversity. The occurrence of mutation is generally associated with low probability. A proper balance between genetic quality and diversity is therefore required within the population in order to support efficient search.

Implementation: We have used C++ programming language to implement out system. The main advantages of C++ include a clean object oriented approach. The following figure describes the flowchart of the system.

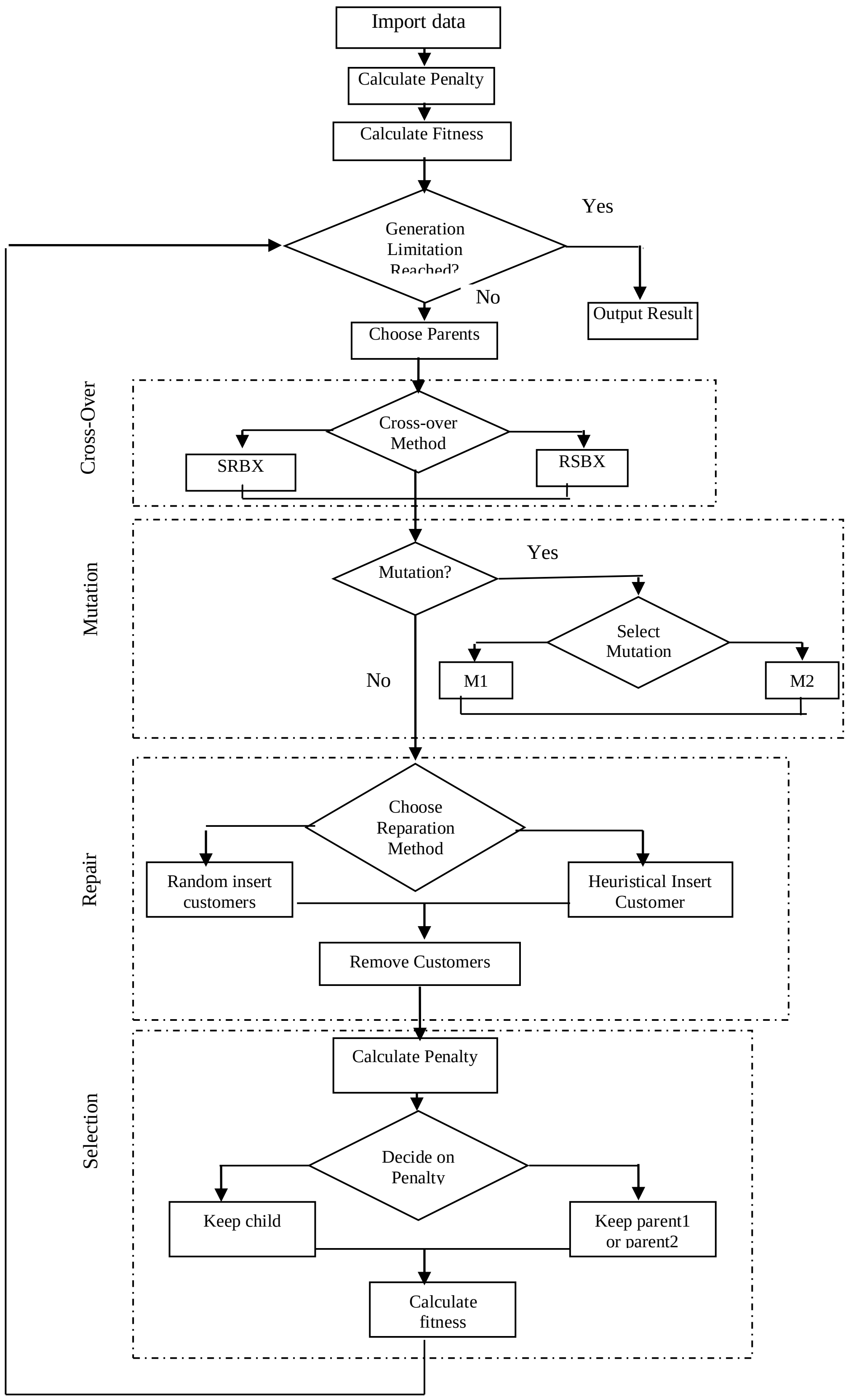

Figure1: Flow Chart of solution of VRP using Genetic Algorithm.

A. Chromosome representation

The individuals of a population in the GA can be seen as an ordered list of artificial chromosomes where every chromosome represents a route a truck is going to take. Each chromosome contains K integers, where K is the number of genes a chromosome holds. A gene itself is and integer as well and represents the number of customer. Example of a solution of 4 trucks with 10 customers.

route1: [2 4 9 10]

route2: [4 6]

route3:[ ]

route4:[3 1 6 7 8]

Route1 is served by truck1 that visits the ordered list from left, starting with customer2, to the right ending at the customer 10 before it goes back do the depot. Trucks that aren't needed in the solution have and empty list.

B. Chromosome implementation

Each set of chromosomes represent one individual, which is one possible solution to the VRP(if all constraints are satisfied then it can be considered as valid solution). Each chromosome represents a Route and is implemented by a Route object. The Route Object stores all the genes (references to customer objects) in an array. The index of the array specifies the position of the customer in the route. All the routes of a solution are stored in an array in the VRP Object, where the index defines the route number. On the top level the VRPManager stores all the VRP objects which define the population. The implementation choice to use an array is optimal for the VRPManager (to hold the VRP objects) and the VRP objects (to hold the routes) as we have a non-changing population and set of routes. For the storage of the genes (customers) a simply linked list world be more suitable then an array as the insertion and removal process could be faster then its currently implemented.

C. Crossovers

In genetic algorithms, crossover is a genetic operator used to vary the programming of a chromosome or chromosomes from one generation to the next. It is an analogy to reproduction and biological crossover, upon which genetic algorithms are based. Both implemented crossovers don't do mutual exchange of genetic material between two parents. They take information from one individual and insert it in the other to create a new child. The probability which crossover method should be used can be configured.

D. Mutation

In genetic algorithms, mutation is a genetic operator used to maintain genetic diversity from one generation of a population of chromosomes to the next. It is analogous to biological mutation. The probability which mutations will take place and if mutation takes place at all can be configured.

E. Reparation

In the reparation process the child first gets checked if it contains some genetic information too much or is missing some. In other words, the process checks which customers are missing on the routes and which ones would be served several times. Customers that are served more then once are removed from the chromosomes that one customer is only present one time. The location from where the duplicated genes are removed is chosen randomly. Customers that are missing need to be re-inserted. Here the heuristic comes into place. the customers are not just inserted in a random location but in a location where they are applicable. This location is found by trying to insert a customer to an existing route in a specific position and checking how much the penalty increase for this route. This process is now applied to all routes, until the route and the position in the route is found where the customer adds the least possible penalty. This step is very time consuming, therefore this method is just used depending on a defined probability. Else the customer is just inserted in a random route at a random position.

F. Penalties

To rate the fitness of a chromosome a special penalty system was integrated. This principle helps to distinguish good routes from bad routes. Different aspects are considered when applying the penalty calculation like the distance of the route or the delay if a customer is served to late. These different penalty operators can be individually adjusted.

G. Child or Parent(s)

After the penalties have been calculated for a new child, the system decides if the child is accepted or not. This process compares the penalties of its parents with the ones from the child. If the child does better, it will be accepted for the next generation. However if the child has a bigger penalty then there will be another selection process, which favors the parents depending on their penalties. In this process also the child has a chance to survive, as it is important for the genetic diversity. If the child wouldn't have a chance to survive in case it has a bigger penalty then its parents, the system would be strictly monotone decrease the overall penalty level of its hole population. This however could make the system stuck in local minimum penalty level from which it couldn't escape any more. Therefore its important to give even a bad child a chance to survive.

H. Fitness

To choose which children from the newly created population will be favored to breed, the fitness of every individual has to be computed. This is done by summing up all the penalties of its chromosomes and using them in the following formula where the max_penalty is the biggest total penalty of one individual found in this generation.

$$\text{Fitness}=100* \frac{\max Penalty - penalty}{\max Penalty}$$

Therefore the fitness is not an absolute measure like the penalty(which can be compared over different generations) but a local measurement for this generation. The Fitness can take values between 0 (which is assigned to the individual with the maximal penalty) up to theoretically 100(which is practice is not reached).

I. Selection process depending on the Fitness

The calculated fitness helps now to select members for the next generation. This is done using the Roulett Wheel Selection method where individuals with a higher fitness are more likely to be selected then others.

III. Results

First we checked our engine with different datafiles. For all datafiles tested, we found sooner or later a valid solution. Some solutions were quite good(compared to published values on the internet) while others were not really satisfying. Here is an example of a solution which is valid, however we can already visually tell that its not optimal as there are some bigger detours which most likely are unnecessary.

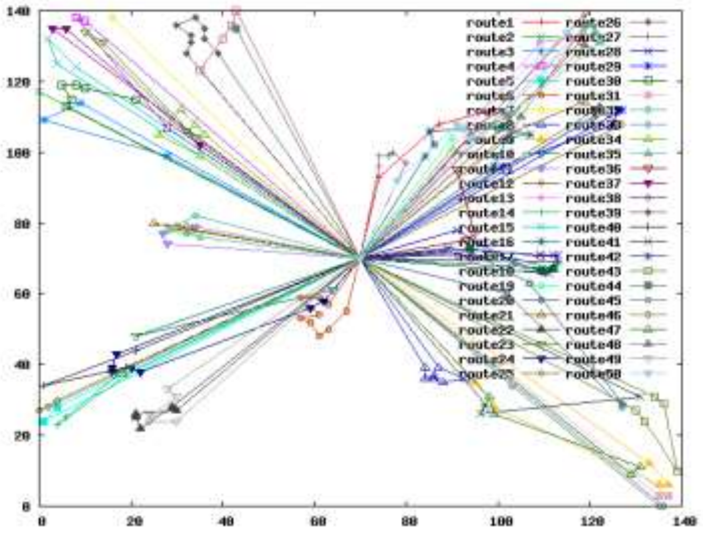

Figure2: Route with 50 trucks.

We used a couple of files [7] to measure how long it takes till the system finds the first valid solution. The following dump lists the filename of the testdata, the traveling distance of all trucks, the time all the trucks together spend on the road and the time it took to find the first valid solution.

```
file: R101.txt ODist 962 OTime 1140 time: 4.68799996376038

file: R102.txt ODist 770 OTime 1085 time: 5.59299993515015

file: R103.txt ODist 768 OTime 942 time: 5.625

file: R104.txt ODist 717 OTime 1021 time: 5.6100001335144

file: R105.txt ODist 799 OTime 1009 time: 6.0939998626709

file: R106.txt ODist 731 OTime 1048 time: 5.60900020599365

file: R107.txt ODist 727 OTime 1076 time: 5.64099979400635
```

IV. System Tuning

We used some heuristic to place missing customers back to the different routes after they disappeared during crossover or mutation. We tried different probabilities on how often the heuristic should be used and measured the different penalties we found as an end solution. We have to set a maximal generation limit which was higher for test runs with lower heuristic probability then with higher probability that every test run took more or less the same time.

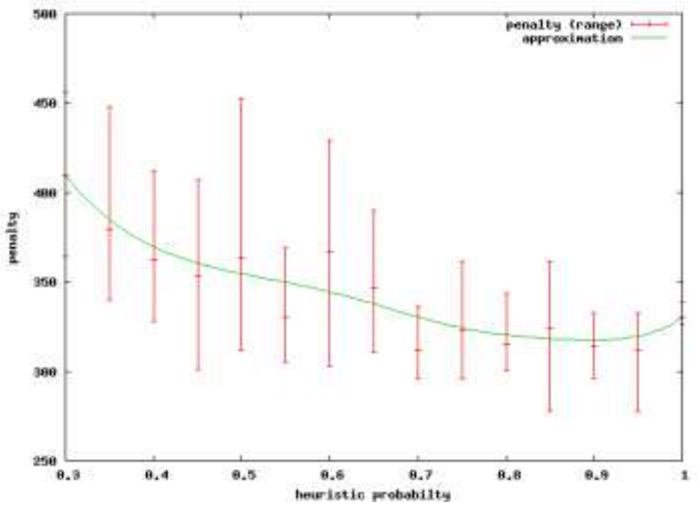

Figure3a: Heuristic probabilities

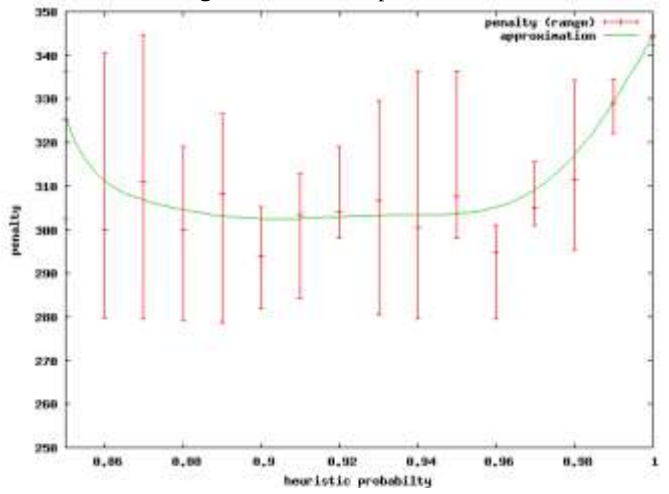

Figure3a: Heuristic probabilities

It can be seen, that with a higher heuristic probability, better results are archived. However if the probability of using heuristics get close to 1, the penalty increases. Therefore we used a finer granulation to see what is going on. It looks like if when the probability gets close to 1, the genetic diversity is not anymore sufficient and we get struck in a local minimum. It is observed that the penalty range (indicated by the error bars) almost collapses where the deviance is much higher else.

Population size: The other important parameter is the generation size. The next figure shows different generations sizes combined with different heuristic probabilities. P30 refers to a population of 30 individuals where p10 represents 10 individuals. R0.3 means that the insertion heuristic was used in 30% of the time.

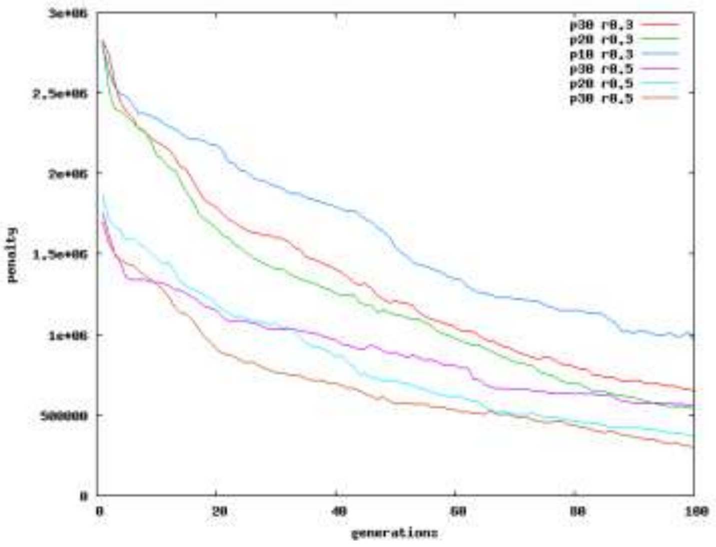

Figure4: population size ½

The first thing that sticks to the eye is, that a higher heuristic probability results in a lower initial penalty and also in a lower end penalty, howeer the difference is not that big any more. The second thing to note is, that the two graphs with a population size of 10 result in the worst end result. Both the population with 20 and 30 individuals are doing quite well. The next graph continues this picture and plots the generations 100 up to 200.

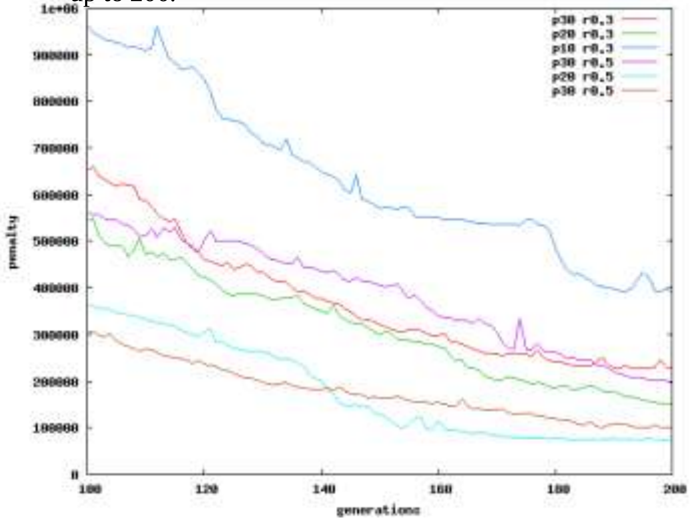

Figure5: population size 2/2

Here now something interesting happens. One would assume that the population size of 30 results in the best end result, however in both cases of the different heuristic probabilities the generation size of 20 finds the best solution. We ran this tests a couple of time and always got the same results.

V. BEST POSSIBLE SOLUTION

We checked for different datafiles if we can get close to the best known solutions and managed to archive the same best result for the file RC208.txt as it is published on the internet. The solutions just uses 1 truck which drives the minimal distance of 328.2.

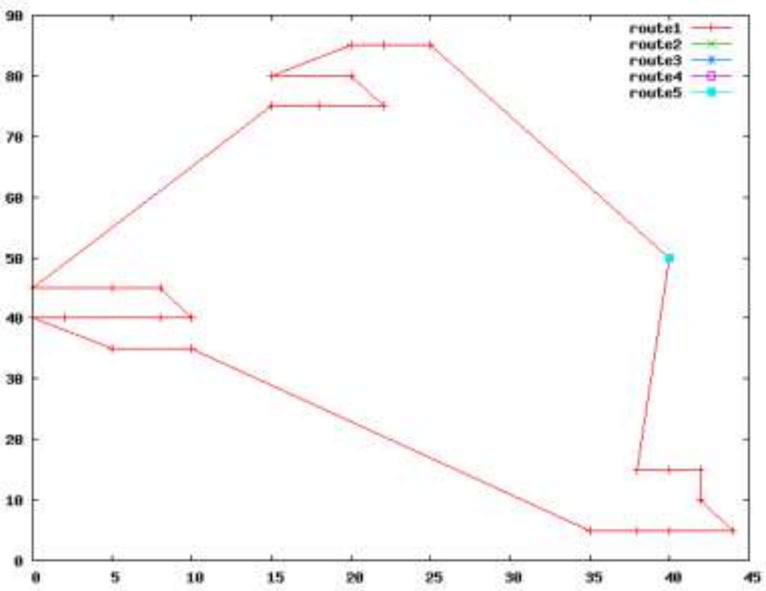

Figure6: Shortest Path.

VI. CONCLUSION

Genetic algorithms provide a very interesting approach to solve problems where an exact method can not be applied. We were a bid disappointed that it took many generations to find a solution, which was not even really good. Therefore we decided to implement some custom insertion heuristic which helps the system to faster approach a good solution. The choice of the crossover method was pretty intuitive and we can't assess if they are good or not. We implemented one that inserts new elements as a subroute using our heuristic method and another one, which does a simple sequence based crossover. We found out that if we put a too high heuristic level, we get on one hand quite fast good results, however in most cases we are unable to get to the best results. Therefore we tuned the system to have a balance between finding fast a solution and limiting the search space.

REFERENCES

[1] M. Fisher. Vehicle routing. Handbooks of Operations Research and Management Science, Chapter 1, 8:1-31,1995

[2] Sergios Theodoridis, Konstantinos Kourtoumbas. Pattern Recognition Second Edition page 582

[3] M.Gendreau, G. Laprte, and J-Y. Potvin. Metaheuristics for the vehicle routing problem. Management Science, 40:1276-1290,1994.

[4] G. Laporte, M. Gendreuau, J-Y. Potvin, and F. Semet. Classical and modern heuristics for the vehicle routing problem. International Transactions in Operational Research, 7:285-300,2000

[5] G. Laporte and F. Semet. Classical heuristics for the vehicle routing problem. Technical Report G-98-54, GERAD, 1999.

[6] The VRP Web. The vrp web, 2004. URL http://neo.lcc.uma.es/radi-aeb/WebVRP/index.html?/results/BestResults.h%tm.

[7] Vehicle Routing Data Sets. Vehicle routing data sets, 2003. URL http://branchandcut.org/VRP/data/

[8] Fogel. D.: An evolutionary approach to the graveling salesman problem. Biological Cybernetics 60.

[9] Rochat, Y., Taillard, E.: Probabilistic diversification and intensification in local search for vehicle routing. J. of Heuristics 1 (1995) 147-167

[10] Croes, G.: A method for solving traveling salesman problems. Operations Research 6. 791-812

[11] Berger, J., Barkaoui, M.: A hybrid genetic algorithm for the capacitated vehicle routing problem. In Cantu-Paz, E., ed.: GECCO03, LNCS 2723, Illinois, Chicago, USA, Springer-Verlag. 646-656

[12] Dantzing, G., Ramster, R.: The truck dispatching problem. Management Science 6(1959) 80-91

AUTHORS PROFILE

Abdul Kadar Muhammad Masum is a graduate of B.Sc and MSc in Computer Science and Engineering from International Islamic University Chittagong, and United International University, Dhaka, Bangladesh, respectively. He has been serving as a faculty member in IIUC since 2005. Now he is an Assistant Professor of Department of Business Administration, International Islamic University Chittagong, Bangladesh. Moreover, he has mentionable research works on Data Mining, E-commerce and E-governance and has a great interest in MIS field. He has also experience in research paper presentation in many International and National Conferences.

Mohammad Shahjalal has received his B. Sc. degree in Electrical and Electronic Engineering from Chittagong University of Engineering and Technology in 2009. Now he is serving as a lecturer in the department of Electrical and Electronic Engineering of International Islamic University Chittagong since February 28, 2010. His research interests focus on Artificial Intelligence, image processing, wireless communication, wireless networking, digital signal processing.

Md. Iqbal Hasan Sarker has received his B.Sc degree in Computer Science & Engineering from Chittagong University of Engineering & Technology (CUET) in 2009. Now he is serving as a Lecturer in Chittagong University of Engineering & Technology (CUET) in the department of Computer Science & Engineering since September 19, 2010. His research interests focus on Artificial Intelligence, Digital Image Processing and Natural Language Processing, Wireless Communication.

Md. Faisal Faruque has received his B.Sc degree in Computer Science & Engineering from International Islamic University Chittagong (IIUC) in 2007. Now he is serving as a Lecturer in University of Information Technology & Sciences (UITS) in the department of Computer Science & Engineering since February 18, 2008. His research interests focus on Artificial Intelligence, Digital Image Processing and Natural Language Processing.