International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Index Copernicus Value (2013): 6.14 | Impact Factor (2013): 4.438

A Steady-State Genetic Algorithm for Traveling Salesman Problem with Pickup and Delivery

Monika Sharma[1], Deepak Sharma[2]

[1]Research Scholar Department of Computer Science and Engineering, NNSS SGI Samalkha, Kurukshetra University Kurukshetra

[2]Research Scholar Department of Computer Science and Engineering, Uttarakhand Technical University, Uttarakhand

Abstract: ***A Steady State Genetic Algorithm (GA) is proposed for the Traveling Salesman Problem with Pickup and Delivery (TSPPD). TSPPD is an extension of the well known Traveling Salesman Problem (TSP). TSPPD is a graph and grouping optimization problem.In this thesis, TSPPD is differentiated by a group of cities as customers, each of them supplying (picking customer) or demanding (delivery customer) a given amount of a single product. The objective is to find out minimum tour length of the route for a capacitated vehicle in order to transport the product from the pickup to the delivery customers. Each city must be visited exactly once and capacity of vehicle should not be violate.In this thesis we have used a pheromone based crossover operator that utilizes both local and global information to construct offspring. In addition, we have also used a local search procedure in the genetic algorithm to accelerate convergence. To selecting parents for crossover and mutation operator to generate feasible offspring, we have binary tournament section method. The results of our algorithm have been tested on benchmark instances and computational results show that we have got comparable results to the optimal results.***

Keywords: Genetic Algorithm, Crossover operator, offspring, Travelling Salesman Problem

1. Introduction

Traveling salesman problem is a combinatorial problem. Basically in TSP, a set of cities and the distance between these cities has to be given. The main objective of TSP is to find out a minimum distance traveled by salesman and each city must be visited exactly once. The problem can be defined by either coordinates of cities or distance matrix that gives the distance between two cities. The minimum path is one that visits each city once and only once. Thus, it is a Hamiltonian path.

Traveling salesman problem with pickups and deliveries (TSPPD) is an extension of traveling salesman problem (TSP). TSPPD is characterized by a set of customers; each customer is associated with two types of quantities of goods to be collected and to be delivered. A vehicle with given capacity is located at the depot. The TSPPD involves that a tour will start and end at the depot and each customer must be visited exactly once. In our algorithm, the problem is defined by a distance matrix that gives distance between two cities. The main objective of TSPPD is to minimize the total tour length without violating the capacity of vehicle.

2. Literature Review

R.Baldacci et al [32] described a two-commodity Traveling Salesman Problem with mixed Deliveries and Collections (TSPDC). In TSPDC, a vehicle located at the depot must be optimally used to serve two set of customer to be delivered and to be collected. The main objective is to minimize the overall tour length and vehicle capacity should not be violate.They present new lower bounds which are obtained from the LP-relaxation of the new formulation by removing some constraints and adding valid inequalities. To improve the lower bound, three class of valid equality are described. The lower bound is computed by cutting-plane based linear programming procedure. The resulting lower bounds are embedded in a branch-and- cut algorithm to solve the TSPDC for optimal solution. The resulting cutting-plane algorithm was applied to three classes of test problems that are taken from the literature and involving problems with up to 261 customers. Hipólito Hernández-Pérez et al [16] proposed a linear programming based heuristic approach for one commodity pickup and delivery traveling salesman problem. Their heuristic procedure has the advantage of not depending on linear programming tool.In the One-Commodity Pickup-and-Delivery Traveling Salesman Problem (1-PDTSP), a finite set of cities is given and the travel distance between two cities is assumed to be given. One of specific city is considered as a vehicle depot while the other cities are identified as customers. Customers are divided into two groups according to the services to be delivery or pickup. There is a unique product to be transported between customers. Each delivery customer requires a given product amount and each pickup customer provides a given product amount. Any amount of the product collected from a pickup customer can be supplied to a delivery customer. It is assumed that the vehicle has a fixed upper-limit capacity nd must start and end the route at the depot. The objective of 1-PDTSP is to minimize the tour length for the vehicle to satisfy the customer requirements without ever violating the capacity of vehicle. The route must be a Hamiltonian tour.

In their approach they combine several operators in a Variable Neighborhood Search scheme. These operators can be grouped into two classes: classical operators for TSP solutions (e.g. 2- opt, Lin-Kernighan, etc) and operators exploiting the load evolution through a route. In addition they also proposed perturbation operators to diversify the search. These operators may create infeasible 1-PDTSP routes, and therefore procedures to restore the feasibility are also designed. G. Vahdati, M. Yaghoubi, M. Poostchi,, M. B. Naghibi S.[13] proposed a new solution for Traveling Salesman Problem (TSP), using genetic algorithm. A

heuristic crossover and mutation operation have been proposed to prevent premature convergence. One of the permanent challenges for GAs is how to deal with premature convergence due to a sudden and fast reduction of search space and also getting stuck in local optima. heuristic crossover, actually uses four pointers, one pair for each parent, which will move clockwise and counterclockwise. These pointers will be evaluated by means of a fitness function, equal to the inverse distance between two cities, and try not to get stuck in local optima. This approach improves the convergent speed towards the global optimal solution. The heuristic mutation will prevent premature convergence, too. The role of adaptive and nonlinear probabilities of crossover and mutation is to solve the low stability and also slow convergent. It must be mentioned that the implementation of algorithm has no complexity. The method includes the results of 30 times implementation of Slandered Algorithms and proposed method. Initial population for all of these methods is equal to 100 Hipólito Hernández-Pérez et al [15] proposed a branch-and-cut algorithm for one commodity pickup and delivery traveling salesman problem and closely related traveling salesman problem. They used a 0-1 integer linear model for finding a optimal solution. In branch-and-cut algorithm follows a branch-and- bound scheme, in which new lower bounds are computed by solving linear program (LP) relaxation of the problem. The relaxation is tightened by adding valid inequities to the current LP, according to the cutting plane approach. They have implemented the algorithm with basic ingredients, where the initial heuristic is based only on the TSP nearest insertion and where the cutting plane does not consider the TSP 2-matching inequalities

In this thesis, we have applied Steady-State Genetic algorithm to solve the TSPPD. In our GA, we implemented pheromone based crossover operator which is used applied to construct the offspring. The pheromone based crossover contains both local and global information. Local information used here includes edge lengths and adjacency relations, whereas global information is stored as pheromone trails. The utilization of global information in crossover can increase the chance to find a better solution, and then improve the performance of GA.

2.1 The Proposed GA for TSPPD

In our proposed GA, path representation is used. After initializing the parameters used in algorithm, the GA starts from a random population of individuals (feasible tours), and iteration until some pre-defined stopping criteria is satisfied. Each individual is evaluated in each iteration (generation) based on its fitness function. During iteration, individuals are probabilistically selected from the population according to the binary tournament selection. Then the selected individuals are recombined or mutated to generate offspring. To accelerate the convergence of GA, a 2-opt local search procedure is used after recombination (crossover).The basic design of our algorithm includes all these steps:

2.1.1 Initial Population Generation

In our algorithm, we have used an adapted nearest neighbor heuristic to generate initial population. We have used iteratively method to generate initial population. During iteration, a city is selected uniformly at random in the individual. Now we will choose the nearest city, according the distance, from the cities that have not appeared in the individual. The procedure is repeated until all cities have appeared in the individual and feasible individual has been generated. The others individuals of population are filled in the same way.

The adapted heuristic is described as follows:

Step 1: Take an empty individual.
Step 2: Select uniformly random city t in the individual and check the feasibility of the city
Step 3: Insert the city into the individual if city is feasible else go to step 2.
Step4: Select the nearest one city according the distance that has not been appeared in the individual.
Step 5: Check the feasibility of the nearest city.
Step 6: If nearest city is feasible then insert it into the individual and go step 4.
Step 7: Procedure is repeated until the entire individuals have been filled.

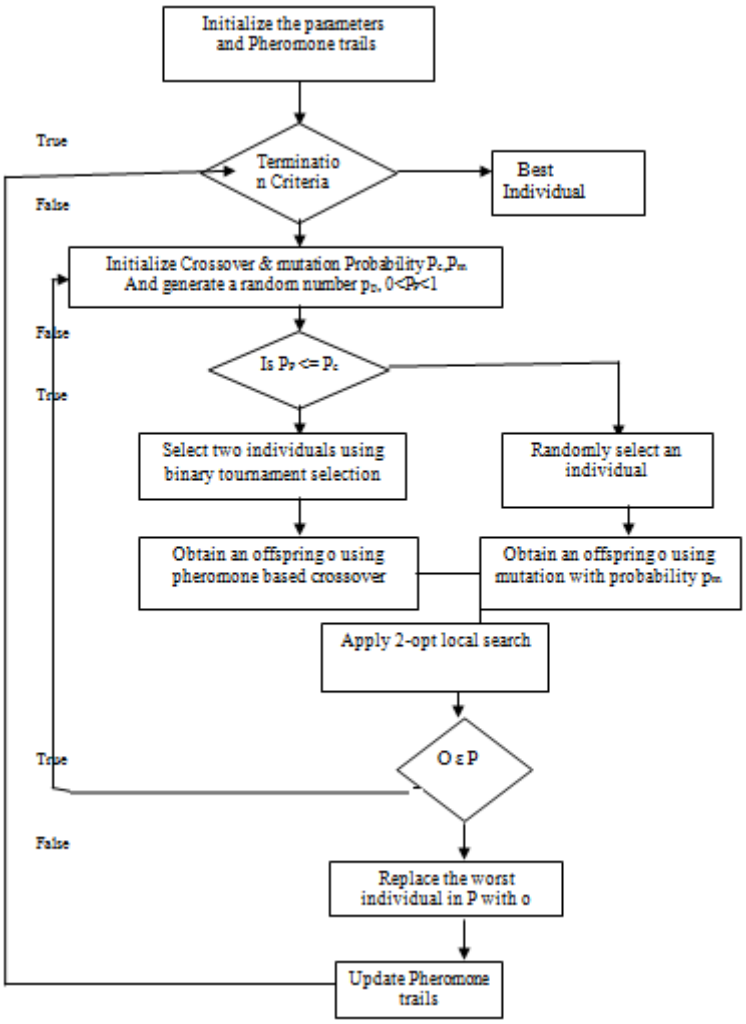


Figure 5.1: Flow chart of Steady-State Genetic Algorithm

2.1.2 Fitness Function

Our fitness function is same as the objective function i.e., the tour length of the individual starting from the depot and ending at the depot. Individual with the minimum tour length consider as a best fit individual.

2.1.3 Selection Method

We have used Binary Tournament Selection method for selecting parents for crossover and mutation operator. We first randomly select two solutions from the population. We then select from these two the least cost solution or best fittest individual to be the first parent P1. This procedure is repeated until we get the second parent P2. These two individual (P1 and P2) are then used for crossover to generate offspring.

2.1.4 Pheromone based Crossover

When generating an offspring, the pheromone-based crossover operator first randomly selects a customer I as the first customer in offspring. Then, the operator will search customer i in the two parents P1 and P2, and identify the immediate predecessors and successors, named neighbors of customer i. The nearest feasible customer j among unvisited customers in the neighbors will be selected as the next customer in offspring. If all customers in neighbors have been visited or all of them cannot be feasibly added to the end of the route, the next customer will be selected according to the following pseudo-random-proportional rule [11].

Step 1: Initially offspring o(k) is empty.
Step 2: Select a random customer i that can be feasibly visited as the first customer.
Step 3: Repeat the step until all customer have been visited
Search customer i in the parents P1 and P2.
if(all the neighbors of customer i have been visited in Parents P1 and P2 or all neighbors cannot be feasibly added in the offspring)

Generate a random decimal fraction p.

Select the next customer j according equation (11).
else
Select the unvisited nearest neighbor as the next customer that can be feasibly added in the offspring.
Step 4: Insert the next customer into the offspring o(k). All customers of offspring will be select
in similar manner.
Step 5: Go to Step 3.

2.1.5 Implementation of Genetic Algorithm

An individual solution in genetic algorithm is represented by a set of parameters. These parameters are called genes which form a chromosome or individual, when structured as a string. The genetic algorithm begins with a population of such chromosomes. The chromosome is

divided in parts: genes which code for properties and has a unique position on the chromosomes.

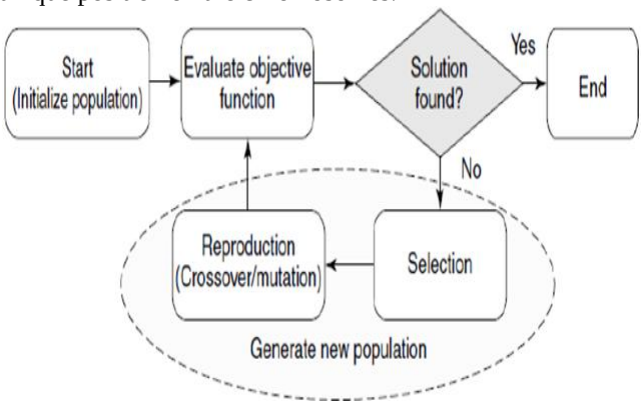


Figure 4.2: showing a simple working of genetic algorithm.

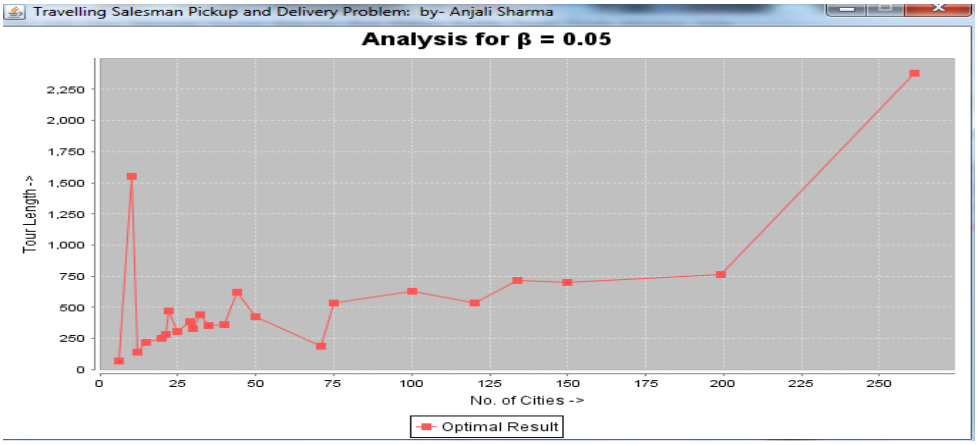
Flow chart basic genetic algorithm iteration

Initially population of individuals is randomly generated to have an unbiased representation of the search space. Each individual has its fitness value which show how individual is much better than other individuals. To find out the fitness of an individual, one may need to convert it first to the phenotype. The fitness may be evaluated then through the interaction of this phenotype withits environment which may also involve the neighborhood of this individual. The module which is used for optionally converting a genotype to phenotype and evaluating the fitness is called evaluation function [10]. After assigning fitness values to each member of the population, a termination criterion is checked. The termination criteria can be a time constraint, computational steps constraint or some value of the fitness which is to be achieved. If a termination criterion is satisfied then algorithm is terminated at once otherwise generate new population by using genetic operators: Selection, crossover and mutation. The selection method select individuals from the current population that having better fitness. This is done to generate fitter individuals in new population in the hope that their offspring will in turn have even higher fitness. After selection method, a recombination process is use to generate offspring, means mixing the genes of two or more individuals (called parents) and from them generating one or more new individuals (called offspring). Each of these offspring will have some genes from one parent and rest from the other.Now the offspring will undergo mutation. Mutation means random change in gene values or information content of a chromosome relevant to the problem under consideration. Both mutation and crossover occur with certain probabilities called mutation rate and crossover rate. Mutation rate decides how many genes of a chromosome will be mutated and crossover rate determines what fraction of old generation will go to the next generation through crossover process [10]. As a result of reproduction, a new population is generated; this is called next generation of the population. The new generation is now evaluated. Termination criterion is tested again and if it is satisfied the algorithm is terminated at once else the genetic algorithm will continue till the termination condition is satisfied. So a simple genetic algorithm works as follows:

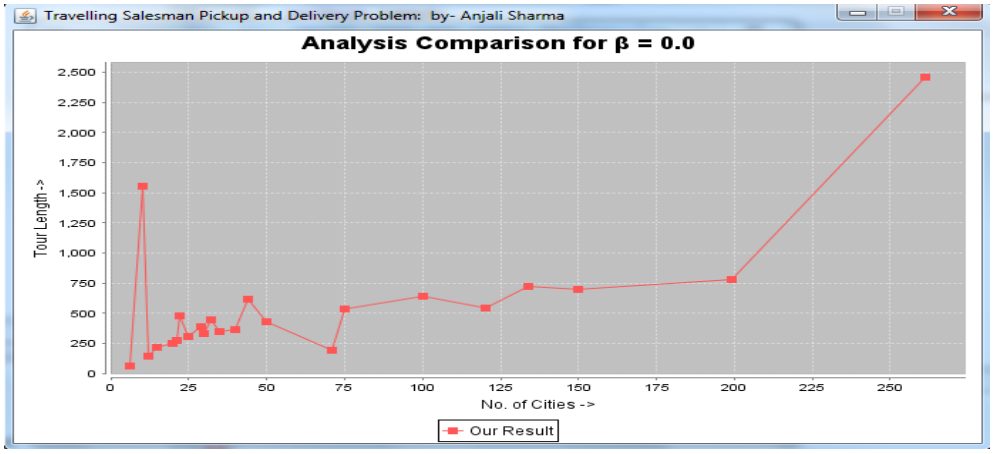
Step 1: Initially generate random population of n individuals.
Step 2: Evaluate the fitness function f(x) of each individual x in the population
Step 3: Repeat the step until n offspring have been created

Select a pair of parent chromosomes from the current population, the probability of selection being an increasing function of fitness. With probability Pc (Crossover-rate), crossover the parents to form new offspring. If no crossover was performed, offspring is an exact copy of parents. With a mutation probability (P chromosome). Mutates new offspring at each locus (position in
Step 4: Replace the current population with new generated population.
Step 5: Go to Step 2.

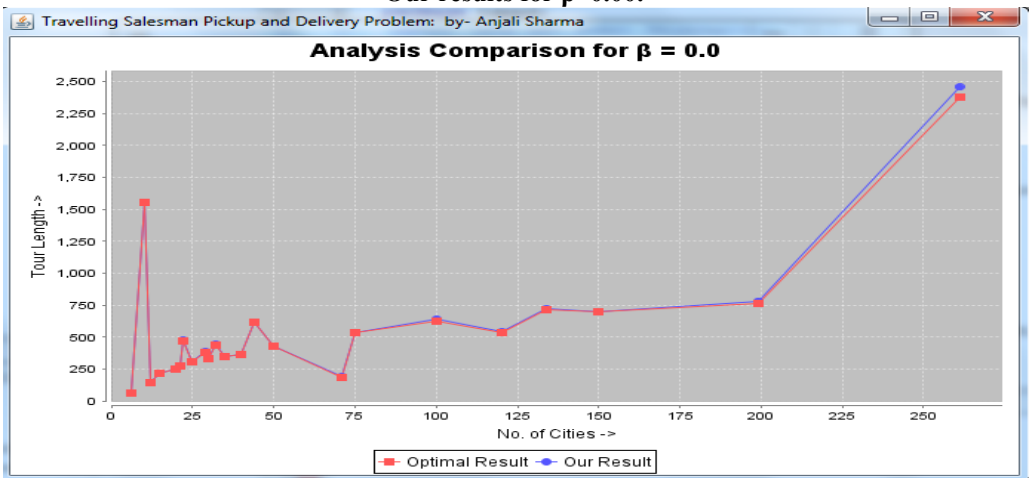
Comparison graph for β= 0.00



Optimal result for β=0.00



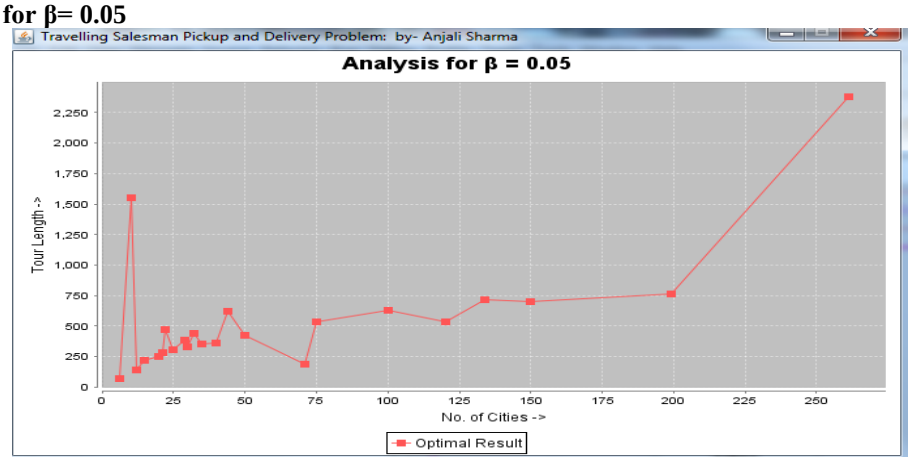
Our results for β=0.00.



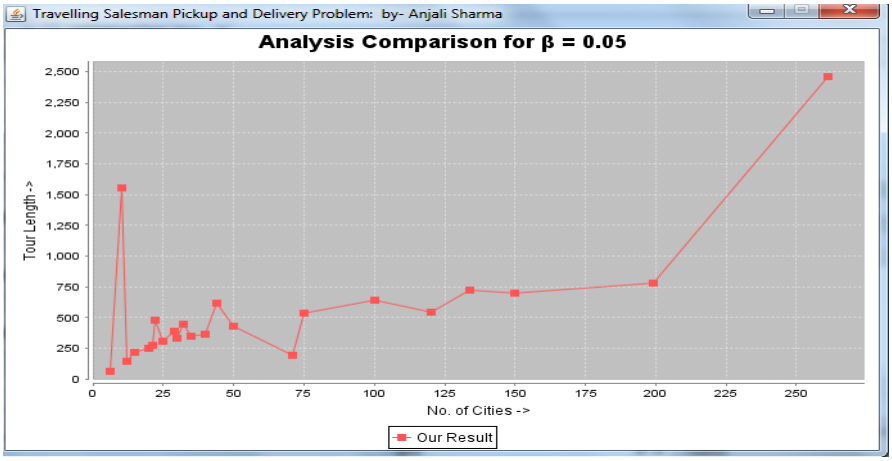
Comparison graph between optimal and our results for β=0.00.

After analyzing the comparison graph for β= 0.00, it shows that, when the number of cities is increases, the tour length of our results is also increases. So graph shows variation for large cities

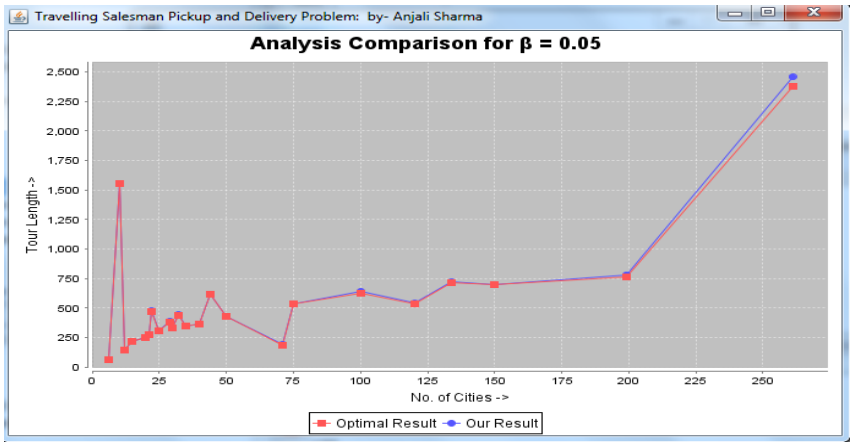
Comparison graph for β= 0.05



Optimal results for β=0.05.



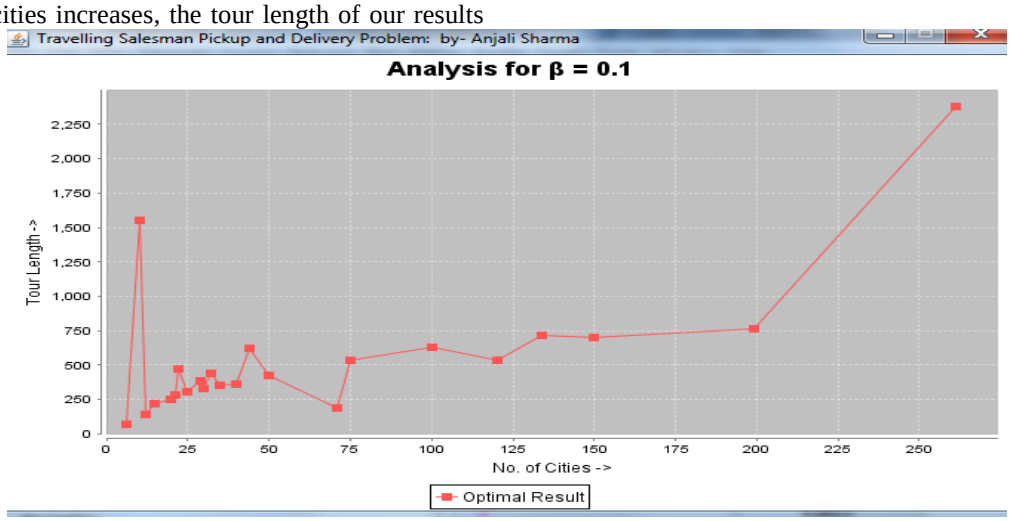
Our results for β=0.05



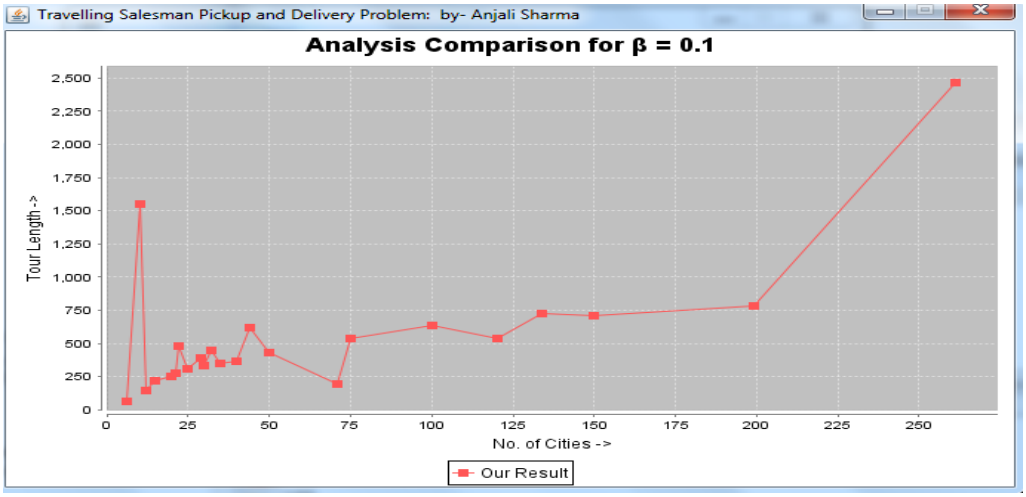
Comparison graph between optimal and our results for β=0.05.

After analyzing the comparison graph for β= 0.05, it is also shows the variations for large cities in the graph. So when the number of cities increases, the tour length of our results is increases

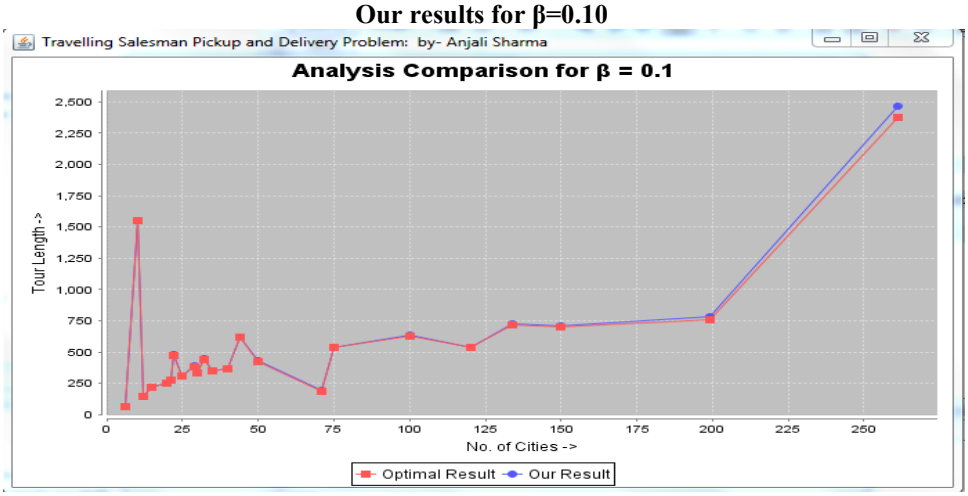
Comparison graph for β= 0.10



Optimal results for β=0.10



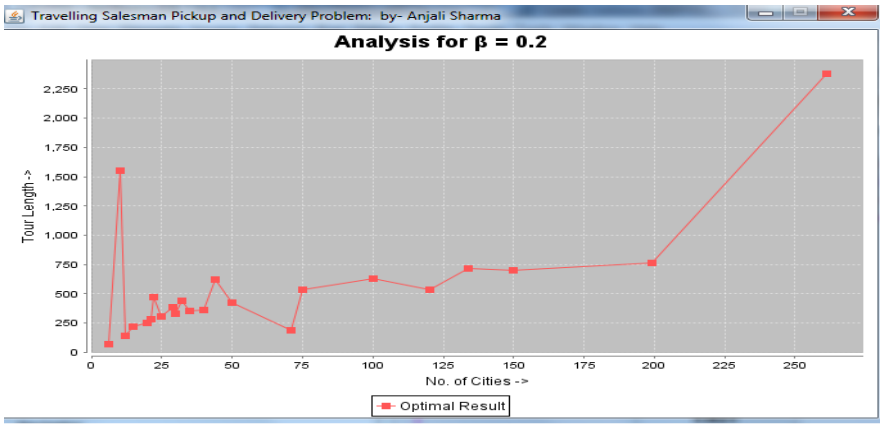
Our results for β=0.10



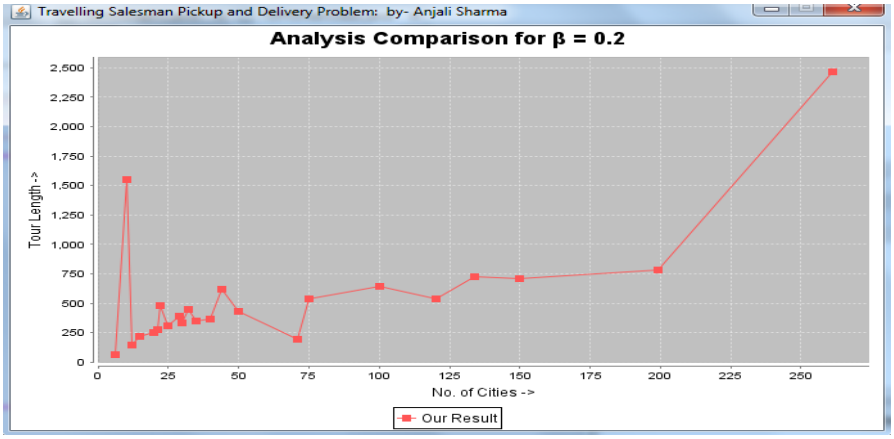
Comparison graph between optimal and our results for β=0.10.

Figure 6.3(C) represents the comparison graph for β= 0.10. It shows that, when the number of cities is increases, the tour length of our results is also increases.

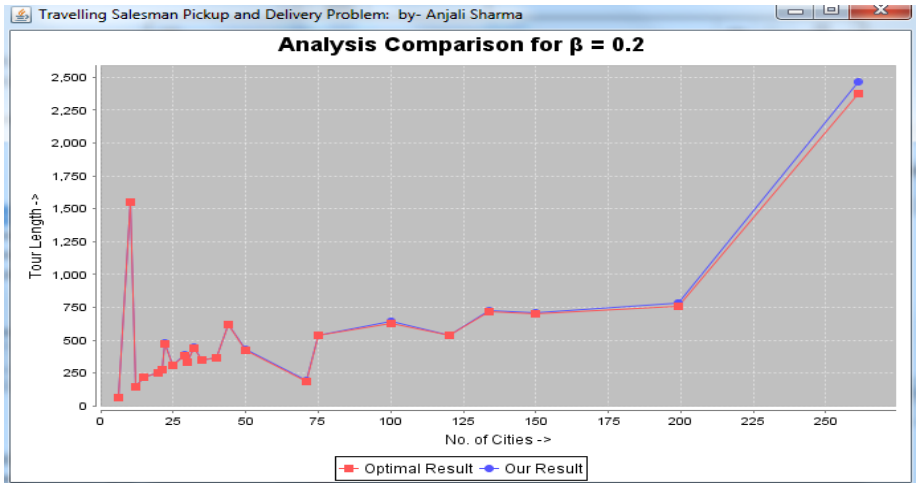
Comparison graph for β= 0.20



Optimal results for β=0.20



Our results for β=0.20



Comparison graph between optimal and our results for β=0.20.

References

[1] Ajith Abraham, Evolutionary Computation, Handbook for Measurement Systems Design, 02143-8, pp. 920-931, 2005

[2] A.E. Eiben, J.E. Smith," Introduction to Evolutionary Computing, Natural Computing Series", Publisher: Springer-Verlag New York, LLC ,2008.

[3] Byung Joo Park, Hyung Rim Choi, Hyun Soo Kim, A Hybrid Genetic Algorithm for Jod Shop Scheduling Problem, computer and industrial engineering vol 45 Issue 4 pp 597 – 613 ,2003

[4] Classification of Meta heuristics, URL:http://www.iiia.csic.es/udt/en/artificialintelligence/combinatorial-optimization-problems

[5] C. Prins, A simple and effective evolutionary algorithm for the Vehicle Routing Problem, Computers and Operations Research, vol 31(12), pp. 1985-2002, 2004.

[6] C Reeves - Handbook of metaheuristics, Springer 2003

[7] Darrell Whitley. A Genetic Algorithm Tutorial, Computer Science Department, Colorado State University, 1994.

[8] Complexity classes, URL: http://en.wikipedia.org/wiki/NP_(complexity)