TSP OPTIMIZATION

(USING PARTICLE SWARM OPTIMIZATION AND ANTS COLONY OPTIMIZATION)

Enrollment Number - 9913103672

Name - Gaurav Chauhan

Name of supervisor(s) - Dr. Mukesh Saraswat

May- 2017

Submitted in partial fulfillment of the Degree of

Bachelor of Technology B. Tech

in

Computer Science Engineering

DEPARTMENT OF COMPUTER SCIENCE ENGINEERING & INFORMATION TECHNOLOGY

JAYPEE INSTITUTE OF INFORMATION TECHNOLOGY, NOIDA

(I)

TABLE OF CONTENTS

(III)

DECLARATION

I GAURAV CHAUHAN hereby declare that this submission is my own work and that, to the best of my knowledge and belief, it contains no material previously published or written by another person nor material which has been accepted for the award of any other degree or diploma of the university or other institute of higher learning, except where due acknowledgment has been made in the text.

Place: Noida Signature:

Date: Name:

Enrollment No:

(IV)

CERTIFICATE

This is to certify that the work titled "**TSP OPTIMIZATION**" submitted by "**GAURAV CHAUHAN**" in partial fulfillment for the award of degree of **Bachelor of Technology (B.Tech)** of Jaypee Institute of Information Technology University, Noida has been carried out under my supervision. This work has not been submitted partially or wholly to any other University or Institute for the award of this or any other degree or diploma.

Signature of Supervisor ……..

Name of Supervisor ……..

Designation ……..

Date ……..

(V)

ACKNOWLEDGEMENT

I would like to express our deepest appreciation to all those who provided me the possibility to complete this project. A special gratitude I give to my faculty Mr. Mukesh Saraswat whose contribution in stimulating suggestions and encouragement helped me to coordinate this project.

Signature of the Student ……..

Name of Student Gaurav Chauhan

Enrollment No. 9913103672

Date of submission ……

(VI)

SUMMARY

The interplay between optimization and machine learning is one of the most important developments in modern computational science. Optimization methods and their applications are proving to be vital in designing algorithms to extract essential knowledge from huge volumes of data. Machine learning, however, is not simply a consumer of optimization technology but a rapidly evolving field that is itself generating new optimization ideas.

The Optimization approaches have enjoyed prominence in machine learning because of their wide applicability and good theoretical properties. The increasing size, complexity, and the variety of machine learning models of today's call for the reassessment of existing assumptions. This project describes the resurgence in novel contexts of established frameworks. It also devotes attention to newer themes as such regularized optimization, robust optimization, gradient and sub gradient methods, splitting techniques, and second-order methods. Many of these techniques draw inspiration from other fields, including operations research, theoretical computer science, and subfields of optimization. This project will enrich the ongoing cross-fertilization between the machine learning community and these other fields, and within the broader optimization community.

In mathematics, computer science and operations research, mathematical optimization, which is also spelled mathematical optimization (alternatively named mathematical programming or simply optimization or optimization), is the selection of a best element from some set of available alternatives.

In simple form , an optimization problem consists of maximizing or minimizing a real function by systematically choosing input values from an allowed set and computing the value of the given function. The generalization of optimization techniques to other formulations comprises a large area of applied mathematics. More generally, optimization includes finding "the best available" values of some objective function given a defined domain (aka input), including a variety of different types of objective functions and different types of domains.

(VII)

LIST OF TABLES AND FIGURES

List of tables

List of figures

CHAPTER 1:
INTRODUCTION

1.1 GENERAL INTRODUCTION

Travelling salesman problem

The Travelling Salesman Problem (often called TSP) is a classic algorithmic problem in the field of computer science. It is focused on optimization. In this context better solution often means a solution that is cheaper. TSP is a mathematical problem. It is most easily expressed as a graph describing the locations of a set of nodes.

Talking about the history, the travelling salesman problem was defined in the 1800s by the Irish Mathematician W. R. Hamilton and by the British mathematician Thomas Kirkman. Hamilton has created a game called Icosian, which was a recreational puzzle based on finding a Hamiltonian cycle. The general form of the TSP were first studied by mathematicians during the 1930s in Vienna and at Harvard, notably by Karl Menger. Menger defines the problem, considers the obvious brute-force algorithm, and observes the non-optimality of the nearest neighbor heuristic.

The Travelling Salesman Problem describes a salesman who must travel between N cities. The order in which he does so is something he does not care about, as long as he visits each one during his trip, and finishes where he was at first. Each city is connected to other close by cities, or nodes, by airplanes, or by road or railway. Each of those links between the cities has one or more weights (or the cost) attached. The cost describes how "difficult" it is to traverse this edge on the graph, and may be given, for example, by the cost of an airplane ticket or train ticket, or perhaps by the length of the edge, or time required to complete the traversal. The salesman wants to keep both the travel costs, as well as the distance he travels as low as possible.

The Traveling Salesman Problem is typical of a large class of "hard" optimization problems that have intrigued mathematicians and computer scientists for years. Most important, it has applications in science and engineering. For example, in the manufacture of a circuit board, it is important to determine the best order in which a laser will drill thousands of holes. An efficient solution to this problem reduces production costs for the manufacturer.

Particle swarm optimization

In field of computer science, **particle swarm optimization** (**PSO**) is a computational method for optimizing a problem by iteratively trying to improve a candidate solution (solution set) with regard to a given measure of quality. In process of solving a problem, is uses a population of candidate solutions, here dubbed particles, and moving these particles (solutions) around in the search-space according to some simple mathematical formulae over the particle's current position and velocity. Each particle's movement is influenced by its local best known position(personal best or pbest), but is also guided toward the best known positions in the search-space, which are updated as better positions are found by other particles(global best or gbest). This is expected to move the swarm toward the best solutions.

PSO is originally featured to Kennedy , Eberhart and Shi and was first intended for simulating the social behaviour, as a designed representation of the movement of organisms in a bird flock or fish school. The algorithm was simplified and it was observed to be performing optimization. The book by Kennedy and Eberhart describes many philosophical aspects of PSO and swarm intelligence. An extensive survey of PSO applications is made by Poli. Recently, a comprehensive review on theoretical and experimental works on PSO has been published by Bonyadi and Michalewicz.

PSO is a metaheuristic as it does not make assumptions about the problem being optimized and can search very large spaces of candidate solutions (possible solutions). However, metaheuristics such as PSO do not guarantee an optimal solution is ever found. Also, PSO does not use the gradient of the problem being optimized, which means PSO

does not require that the optimization problem be differentiable as is required by classic optimization methods such as gradient descent and quasi-newton methods.

Ant colony optimisation

The ant colony optimization algorithm called as ACO, is another probabilistic based technique for solving computational problems as optimization which can be reduced to finding good paths through graphs. This algorithm is a member of ant colony algorithms family, in swarm intelligence methods, the first algorithm was aiming to search for an optimal path in a graph; based on the behavior of ants seeking a path between their colony/position and a source of food. The original idea has since enhanced and diversified to solve a wider class of Numerical problems, and as a result, several problems have emerged and solved, drawing on various aspects of the behavior of ants.

1.2 Problem Statement

To understand what the traveling salesman problem (TSP) is, and why it's so problematic, let's briefly go over a classic example of the problem. Imagine we are a salesman and we have been given a map like the one opposite. On it we can see that the map contains a total of 20 locations and we are told it's our job to visit each of these locations to make a sell. Before we set off on our journey we will probably want to plan a route so we can minimize our travel time. Fortunately, humans are pretty good at this, we can easily work out a reasonably good route without needing to do much more than glance at the map. However, when we've found a route that we believe is optimal, how can we test if it's really the optimal route? Well, in short, we can't - at least not practically. To understand why it's so difficult to prove the optimal route let's consider a similar map with just 3 locations instead of the original 20. To find a single route, we first have to choose a starting location from the three possible locations on the map. Next, we'd have a choice of 2 cities for the second location, then finally there is just 1 city left to pick to complete our route. This would mean there are 3 x 2 x 1 different routes to pick in total. That means, for this example, there are only 6 different routes to pick from. So for this case of just 3 locations it's reasonably trivial to calculate each of those 6 routes and find the shortest. The number of possible routes is a factorial of the number of locations to visit, and trouble

with factorials is that they grow in size remarkably quick! For example, the factorial of 10 is 3628800, but the factorial of 20 is a gigantic, 2432902008176640000. So going back to our original problem, if we want to find the shortest route for our map of 20 locations we would have to evaluate 2432902008176640000 different routes! Even with modern computing power this is terribly impractical, and for even bigger problems, it's close to impossible.

1.3 Empirical Study

1.3.1 Empirical Study of Particle Swarm Optimization

Through cooperation and competition among the population, population-based optimization approaches often can find very good solutions efficiently and effectively. Most of the population based search approaches are motivated by evolution as seen in nature. Four well-known examples are genetic algorithms, evolutionary programming, evolutionary strategies and genetic programming. Particle swarm optimization (PSO), on the other hand, is motivated from the simulation of social behavior. Nevertheless, they all work in the same way, that is, updating the population of individuals by applying some kinds of operators according to the fitness information obtained from the environment so that the individuals of the population can be expected to move towards better solution areas.

The PSO algorithm was first introduced by Eberhart and Kennedy. Instead of using evolutionary operators to manipulate the individuals, like in other evolutionary computational algorithms, each individual in PSO flies in the search space with a velocity which is dynamically adjusted according to its own flying experience and its companions' flying experience. Each individual is treated as a volume-less particle (a point) in the D-dimensional search space. The ith particle is represented as Xi = (xi1, xi2, ... , xiD). The best previous position (the position giving the best fitness value) of the ith particle is recorded and represented as Pi = (pi1, pi2, ... , piD). The index of the best particle among all the particles in the population is represented by the symbol g. The rate of the position change (velocity) for particle i is represented as Vi to the following equation: -- (vi1, vi2, .. . , viD). The particles are manipulated according to the following equation:

$$v_{id} = v_{id} + c_1 * rand( ) * (p_{id} - x_{id}) + c_2 * Rand( ) * (p_{gd} - x_{id}) \quad (1a)$$

$$x_{id} = x_{id} + v_{id} \quad (1b)$$

where cI and c2 are two positive constants, and rand() and Rand() are two random functions in the range [0,1].

1.3.2 Empirical Study Of Ant Colony Optimization

In the natural world, ants of some species (initially) wander randomly, and upon finding food return to their colony while laying down pheromone trails. If other ants find such a path, they are likely not to keep travelling at random, but instead to follow the trail, returning and reinforcing it if they eventually find food.

Over time, however, the pheromone trail starts to evaporate, thus reducing its attractive strength. The more time it takes for an ant to travel down the path and back again, the more time the pheromones have to evaporate. A short path, by comparison, gets marched over more frequently, and thus the pheromone density becomes higher on shorter paths than longer ones. Pheromone evaporation also has the advantage of avoiding the convergence to a locally optimal solution. If there were no evaporation at all, the paths chosen by the first ants would tend to be excessively attractive to the following ones. In that case, the exploration of the solution space would be constrained. The influence of pheromone evaporation in real ant systems is unclear, but it is very important in artificial systems.

The overall result is that when one ant finds a good (i.e., short) path from the colony to a food source, other ants are more likely to follow that path, and positive feedback eventually leads to all the ants following a single path. The idea of the ant colony algorithm is to mimic this behavior with "simulated ants" walking around the graph representing the problem to solve.

Ant colony optimization algorithms have been applied to many combinatorial optimization problems, ranging from quadratic assignment to protein folding or routing

vehicles and a lot of derived methods have been adapted to dynamic problems in real variables, stochastic problems, multi-targets and parallel implementations. It has also been used to produce near-optimal solutions to the travelling salesman problem. They have an advantage over simulated annealing and genetic algorithm approaches of similar problems when the graph may change dynamically; the ant colony algorithm can be run continuously and adapt to changes in real time. This is of interest in network routing and urban transportation systems.

The first ACO algorithm was called the ant system[12] and it was aimed to solve the travelling salesman problem, in which the goal is to find the shortest round-trip to link a series of cities. The general algorithm is relatively simple and based on a set of ants, each making one of the possible round-trips along the cities. At each stage, the ant chooses to move from one city to another according to some rules:

1.It must visit each city exactly once;
2.A distant city has less chance of being chosen (the visibility);
3.The more intense the pheromone trail laid out on an edge between two cities, the greater the probability that that edge will be chosen;
4.Having completed its journey, the ant deposits more pheromones on all edges it traversed, if the journey is short;
5.After each iteration, trails of pheromones evaporate.

1.4 Approach to problem

PSO Algorithm for TSP

we will use the most direct way to denote TSP-path presentation. For example, path 4-2-1-3-4 can be denoted as (4, 2, 1, 3) or (2, 1, 3, 4) and it is referred as a chromosome. Every chromosome is regarded as a validate path. (all paths should be considered as a ring, or closed path). Fitness function is the only standard of judging whether an individual is “good” or not. We take the reciprocal of the length of each path as the fitness function. Length the shorter, fitness values the better. The fitness function is defined as following formula:

$$f(Si) = \Big/ \sum_{i=1}^{N} d[C_{n(i)}, C_{n(i+1) \bmod N}]$$

In order to verify the proposed algorithm is useful for the TSP, the experiment test we select 10 TSP test cases: berlin52, kroA100, kroA200, pr299, rd400, ali535, d657, rat783, u1060 and u1432. All experiments are performed on Intel Core(TM)2 Duo CPU 2.26GHz/4G RAM Laptop. In the experiments all test cases were chosen from TSPLIB And the optimal solution of each test case is known in the website.

We list the test cases' optimal solutions and compared with traditional genetic algorithm and PSO algorithm, the comparison results shown in Table 2. The comparison results demonstrate clearly the efficiency of our algorithm. Note that for the 10 test cases the optimum was found in all ten runs. The number of cities in these test cases varies from 52 to 1432.

Table 2: Optimal results comparison

Test Cases	*Optimal in TSPLIB*	*GA*	*PSO*
Berlin52	7542	7542	7542
kroA100	21282	21315	21310
kroA200	29368	30168	29968
Pr299	48191	48568	48540
Rd400	15281	15135	15135
Ali535	202310	242310	231120
D657	48912	50912	50612
Rat783	8806	8965	8905
U1060	224094	279094	269908
U1432	152970	182780	177890

Ant colony optimization for TSP

The ACO is developed according to the observation that real ants are capable of finding the shortest path from a food source to the nest without using visual cues. To illustrate how the ‘‘real” ant colony searches for the shortest path, an example from will be introduced for better comprehension. In Fig. 1(a), suppose A is the food source and E is the nest. The goal of the ants is to bring the food back to their nest. Obviously, the shorter paths have advantage compared with the longer ones. Suppose that at time t = 0

there are 30 ants at point B (and 30 at point D). And at this moment there is no pheromone trail on any segments. So the ants randomly choose their path with equal probability. Therefore, on the average 15 ants from each node will go toward H and 15 toward C (Fig. 1(b)). At t = 1 the 30 new ants that come to B from A find a trail of intensity, 15 on the path that leads to H, laid by the 15 ants that went that way from B, and a trail of intensity 30 on the path to C, obtained as the sum of the trail laid by the 15 ants that went that way from B and by the 15 ants that reached B coming from D via C (Fig. 1(c)). The probability of choosing a path is therefore biased, so that the expected number of ants going toward C will be double of those going toward H: 20 versus 10, respectively. The same is true for the new 30 ants in D which come from E. This process continues until all of the ants will eventually choose the shortest path.

Given an n-city TSP with distances dij, the artificial ants are distributed to these n cities randomly. Each ant will choose the next to visit according to the pheromone trail remained on the paths just as mentioned in the above example. However, there are two main differences between artificial ants and real ants: (1) the artificial ants have ‘‘memory”; they can remember the cities they have visited and therefore they would not select those cities again. (2) The artificial ants are not completely ‘‘blind”; they know the distances between two cities and prefer to choose the nearby cities from

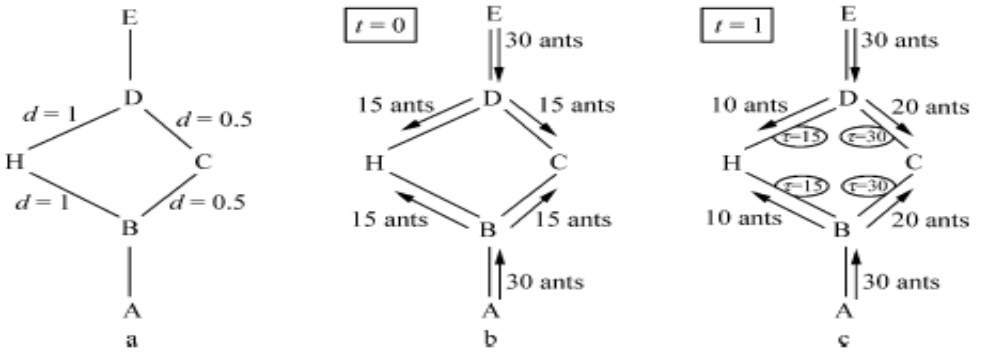

Fig. 1. An example with artificial ants [21]. (a) The initial graph with distances. (b) At time $t = 0$ there is no trail on the graph edges; therefore, ants choose whether to turn right or left with equal probability. (c) At time $t = 1$ trail is stronger on shorter edges, which are therefore, in the average, preferred by ants.

Therefore, the probability that city j is selected by ant k to be visited after city i could be written as follows:

$$p_{ij}^{k} = \begin{cases} \dfrac{[\tau_{ij}]^{\alpha} \cdot [\eta_{ij}]^{\beta}}{\sum_{s \in allowed_k} [\tau_{is}]^{\alpha} \cdot [\eta_{is}]^{\beta}} & j \in allowed_k \\ 0 & otherwise \end{cases}$$

where sij is the intensity of pheromone trail between cities I and j, a the parameter to regulate the influence of sij, gij the visibility of city j from city i, which is always set as 1/dij (dij is the distance between city i and j), b the parameter to regulate the influence of gij and allowed k the set of cities that have not been visited yet, respectively.

1.5 Support for Novelty/ significance of problem

The origins of the travelling salesperson problem are unclear. A handbook for travelling salesmen from 1832 mentions the problem and includes example tours through Germany and Switzerland, but contains no mathematical treatment.

The travelling salesperson problem was mathematically formulated in the 1800s by the Irish mathematician W.R. Hamilton and by the British mathematician Thomas Kirkman. Hamilton's Icosian Game was a recreational puzzle based on finding a Hamiltonian cycle. The general form of the TSP appears to have been first studied by mathematicians during the 1930s in Vienna and at Harvard, notably by Karl Menger.

In the 1950s and 1960s, the problem became increasingly popular in scientific circles in Europe and the USA after the RAND Corporation in Santa Monica offered prizes for steps in solving the problem. Notable contributions were made by George Dantzig, Delbert Ray Fulkerson and Selmer M. Johnson from the RAND Corporation, who expressed the problem as an integer linear program and developed the cutting plane method for its solution. They wrote what is considered the seminal paper on the subject in which with these new methods they solved an instance with 49 cities to optimality by constructing a tour and proving that no other tour could be shorter. Dantzig, Fulkerson and Johnson, however, speculated that given a near optimal solution we may be able to find optimality or prove optimality by adding a

small amount of extra inequalities (cuts). They used this idea to solve their initial 49 city problem using a string model. They found they only needed 26 cuts to come to a solution for their 49 city problem.

In the following decades, the problem was studied by many researchers from mathematics, computer science, chemistry, physics, and other sciences. In the 1960s however a new approach was created, that instead of seeking optimal solutions, one would produce a solution whose length is provably bounded by a multiple of the optimal length, and in doing so create lower bounds for the problem; these may then be used with branch and bound approaches. One method of doing this was to create a minimum spanning tree of the graph and then double all its edges, which produces the bound that the length of an optimal tour is a most twice the weight of a minimum spanning tree.

Christofides made a big advance in this approach of giving an approach for which we know the worst-case scenario. His algorithm given in 1976, at worst is 1.5 times longer than the optimal solution. As the algorithm was so simple and quick, many hoped it would give way to a near optimal solution method. However, until 2011 when it was beaten by less than a billionth of a percent, this remained the method with the best worst-case scenario.

Richard M. Karp showed in 1972 that the Hamiltonian cycle problem was NP-complete, which implies the NP-hardness of TSP. This supplied a mathematical explanation for the apparent computational difficulty of finding optimal tours.

Great progress was made in the late 1970s and 1980, when Grötschel, Padberg, Rinaldi and others managed to exactly solve instances with up to 2392 cities, using cutting planes and branch-and-bound.

1.6 Tabular comparison of other existing approaches

Method	Time (sec)	Fitness Function	Count	Min
1)Exhaustive	426,214	2,627	1	Y
2)Back Tracking	21,549	2,627	1	Y
3)Random Search	0,019	3,438	20	Y
4)Greedy	0,020	2,627	1	Y
5)Hill Climbing	0,005	2,627	10	Y
6)Simulated Annealing	0,016	2,914	20	N
7)Tabu Search	0,385	2,627	2	Y
8) Ant Colony	1,092	2,627	1	Y
9)Genetic Search	1,337	2,627	2	Y
10)Particle Swarms	1,753	2,627	3	Y

Table of results of calculation

The best method for the solution of travel salesman problem was searched. From the ten used methods the Ant Colony and Greedy algorithm were found to be the best. The speed of calculation can be an important parameter when the shortest way is searched at on line access.

Chapter-2
Literature Survey

2.1 Summary of papers studied

Summary of research papers: Paper-1

- Title :Solve Traveling Salesman Problem Using Particle Swarm Optimization Algorithm

- **Authors**

- Xuesong Yan
- Can Zhang
- Wei Li
- Hanmin Liu

- School of Computer Science, China University of Geosciences Wuhan, Hubei 430074, China

- Wuhan Institute of Ship Building Technology Wuhan, Hubei 430050, China

- **Year of Publication:** 2013

- **Publishing Details:**

- IJCSI International Journal of Computer Science Issues, Vol. 9, Issue 6, No 2, November 2012

- Web Link

https://pdfs.semanticscholar.org/d4d3/fa4eadc8b3c4b5989960688a3f2833c984cf.pdf

The traveling salesman problem (TSP) is one of the most widely studied NP-hard combinatorial optimization problems and traditional genetic algorithm trapped into the local minimum easily for solving this problem. Particle Swarm Optimization (PSO) algorithm was developed under the inspiration of behavior laws of bird flocks, fish schools and human communities. Compared with the genetic algorithm, PSO algorithm has high convergence speed. In this paper, aim at the disadvantages of genetic algorithm like being trapped easily into a local optimum, we use the PSO algorithm to solve the TSP and the experiment results show the new algorithm is effective for this problem.

Summary of research papers: Paper-2

- Title

Solving the Travelling Salesman Problem Using the Ant Colony Optimization

- **Authors**

- Ivan Brezina Jr.
- Zuzana Čičková

- **Year of Publication:**

- 2011

- **Publishing Details:**

- Management Information Systems, Vol. 6 (2011), No. 4, pp. 010-014

- **Web Link**

- http://www.ef.uns.ac.rs/mis/archive-pdf/2011%20-%20No4/MIS2011_4_2.pdf

In this paper, we study a possibility of solving the well-known Travelling Salesman Problem (TSP), which ranges among NP-hard problems, and offer a theoretical overview of some methods used for solving this problem. We discuss the Ant Colony Optimization (ACO), which belongs to the group of evolutionary techniques and presents the approach used in the application of ACO to the TSP. We study the impact of some control parameters by implementing this algorithm. The quality of the solution is compared with the optimal solution.

2.2 Integrated summary

Paper-1

Solve Traveling Salesman Problem Using Particle Swarm Optimization Algorithm

The traveling salesman problem (TSP) is one of the most widely studied NP-hard combinatorial optimization problems and traditional genetic algorithm trapped into the local minimum easily for solving this problem. Particle Swarm Optimization (PSO) algorithm was developed under the inspiration of behavior laws of bird flocks, fish schools and human communities. Compared with the genetic algorithm, PSO algorithm has high convergence speed. In this paper, aim at the disadvantages of genetic algorithm like being trapped easily into a local optimum, we use the PSO algorithm to solve the TSP and the experiment results show the new algorithm is effective for this problem.

Particle Swarm Optimization (PSO) algorithm was an intelligent technology first presented in 1995 by Eberhart and Kennedy, and it was developed under the inspiration of behavior laws of bird flocks, fish schools and human communities. If we compare PSO with Genetic Algorithms (GAs), we may find that they are all maneuvered on the basis of population operated. But PSO doesn't rely on genetic operators like selection operators, crossover operators and mutation operators to operate individual, it optimizes the population through information exchange among individuals. PSO achieves its optimum solution by starting from a group of random solution and then searching repeatedly.

PSO is a meta-heuristic as it makes few or no assumptions about the problem being optimized and can search very large spaces of candidate solutions. However, meta-heuristics such as PSO do not guarantee an optimal solution is ever found. More specifically, PSO does not use the gradient of the problem being optimized, which means PSO does not require that the optimization problem be differentiable as is required by classic optimization methods such as gradient descent and quasi-Newton methods. PSO can therefore also be used on optimization problems that are partially irregular, noisy, change over time, etc. This paper improves the disadvantages of

standard PSO being easily trapped into a local optimum and proposed a new algorithm which proves to be more simply conducted and with more efficient global searching capability, then use the new algorithm for engineering optimization field.

Integrated Summary:

Paper-2

Solving the Travelling Salesman Problem Using the Ant Colony Optimization

Travelling salesman problem (TSP) consists of finding the shortest route in complete weighted graph *G* with *n* nodes and *n(n-1)* edges, so that the start node and the end node are identical and all other nodes in this tour are visited exactly once. The most popular practical application of TSP are: regular distribution of goods or resources, finding of the shortest of costumer servicing route, planning bus lines etc., but also in the areas that have nothing to do with travel routes. Here are some of them as an illustration:

- Drilling holes for electrical circuits for flat machines;
- Starlight interferometer satellite positioning;
- Applications in crystallography;
- Chain diagram optimization;
- Computer motherboard components layout;
- Industrial robot control;

Ant colony optimization (ACO) belongs to the group of metaheuristic methods. The idea was published in the early 90s for the first time. The base of ACO is to simulate the real behaviour of ants in nature. The functioning of an ant colony provides indirect communication with the help pheromones, which ants excrete. Pheromones are chemical substances which attract other ants searching for food. The attractiveness of a given path depends on the quantity of pheromones that the ant feels. Pheromones excretion is governed by some rules and has not always the same intensity. The quantity of pheromones depends on the attractiveness of the route. The use of more attractive

route ensures that the ant exudes more pheromones on its way back and so that path is more also attractive for other ants. The important characteristic of pheromones is evaporation. This process depends on the time. When the way is no longer used, pheromones are more evaporated and the ants begin to use other paths.

What is important for ACO algorithm the moving of ants. This motion is not deterministic, but it has stochastic character, so the ants can find the path, which is firstly unfavourable, but which is ultimately preferable for food search. The important characteristic is that a few individuals continuously use non-preferred path and look for another best way.

ACO was formulated based on experiments with double path model, where the quantification was made similar to Monte Carlo method. The base of this simulation was two artificial connections between the anthill and a food source. The simulation demonstrated that ants are able to find the shorter these two paths. A significant impact of this simulation was to quantify the behaviour of ants.

For practical use of ACO, it was necessary to project virtual ants. It was important to set their properties. These properties help virtual ants to scan the graph and find the shortest tour. Virtual ants do not move continuously; they move in jumps, which means that, after a time unit, they will always be in another graph node. The absolved path is saved in ant memory. The created cycles are detected in ant memory. In the next tour, the ant decides on the base of pheromones power. Just because the property of pheromone evaporation, pheromones on shortest edges are stronger, because of the fact that the ant goes across these edges faster.

Chapter 3:
Analysis, Design and Modeling

3.1 Overall description

The **Travelling Salesman Problem** (often called **TSP**) is a classic algorithmic problem in the field of computer science. It is focused on optimization. In this context better solution often means a solution that is cheaper. TSP is a mathematical problem. It is most easily expressed as a graph describing the locations of a set of nodes.

The general form of the TSP appears to have been first studied by mathematicians during the 1930s in Vienna and at Harvard, notably by Karl Menger. Menger defines the problem, considers the obvious brute-force algorithm, and observes the non-optimality of the nearest neighbour heuristic.

A basic variant of the PSO algorithm works by having a population (called a swarm) of candidate solutions (called particles). These particles are moved around in the search-space according to a few simple formulae. The movements of the particles are guided by their own best known position in the search-space as well as the entire swarm's best known position. The process is repeated and by doing so it is hoped, but not guaranteed, that a satisfactory solution will eventually be discovered. Formally, let f : R n → R be the cost function which must be minimized. The function takes a candidate solution as argument in the form of a vector of real numbers and produces a real number as output which indicates the objective function value of the given candidate solution. The gradient of f is not known. The goal is to find a solution a for which f (a) ≤ f (b) for all b in the search-space, which would mean a is the global minimum. Maximization can be performed by considering the function h = − f instead.

For example, path 4 -2-1-3-4 can be denoted as (4, 2, 1, 3) or (2, 1, 3, 4) and it is referred as a chromosome. Every chromosome is regarded as a validate path. Fitness function is the only standard of judging whether an individual is “good” or not.We take the

reciprocal of the length of each path as the fitness function. Length the shorter, fitness values the better.

Ant as a single individual has a very limited effectiveness. But as a part of a well-organized colony, it becomes one powerful agent, working for the development of the colony. The ant lives for the colony and exists only as a part of it. Each ant is able to communicate, learn, cooperate, and all together they are capable of develop themselves and colonise a large area. The self organizing principles they are using allow a highly coordinated behavior of the colony.

Ants communicate to one another by laying down pheromones along their trails, so where ants go within and around their ant colony is a stigmergic system. In many ant species, ants walking from or to a food source, deposit on the ground a substance called *pheromone*. Other ants are able to smell this pheromone, and its presence influences the choice of their path, that is, they tend to follow strong pheromone concentrations. The pheromone deposited on the ground forms a pheromone trail, which allows the ants to find good sources of food that have been previously identified by other ants.

3.2 Functional requirements

a) Input file:-

The program will read data from an external file.

b) *IDE:-*

A IDE like Netabeans ,Eclipse is requires as the program uses external libraries which can be easily imported using IDE.

c) Library File:-

The program will require to include external library file for csv file operations.

3.3) Non Functional requirements

a) Portability:-

The program is 100% portable means it is independent of OS used.

b) Usability:-

The program is easy to use and can be handled by any user with minimal instructions.

c) Reliability:-

The program is operable in all lightning conditions with plain background.

3.3 Design Diagrams

Particle Swarm Optimization

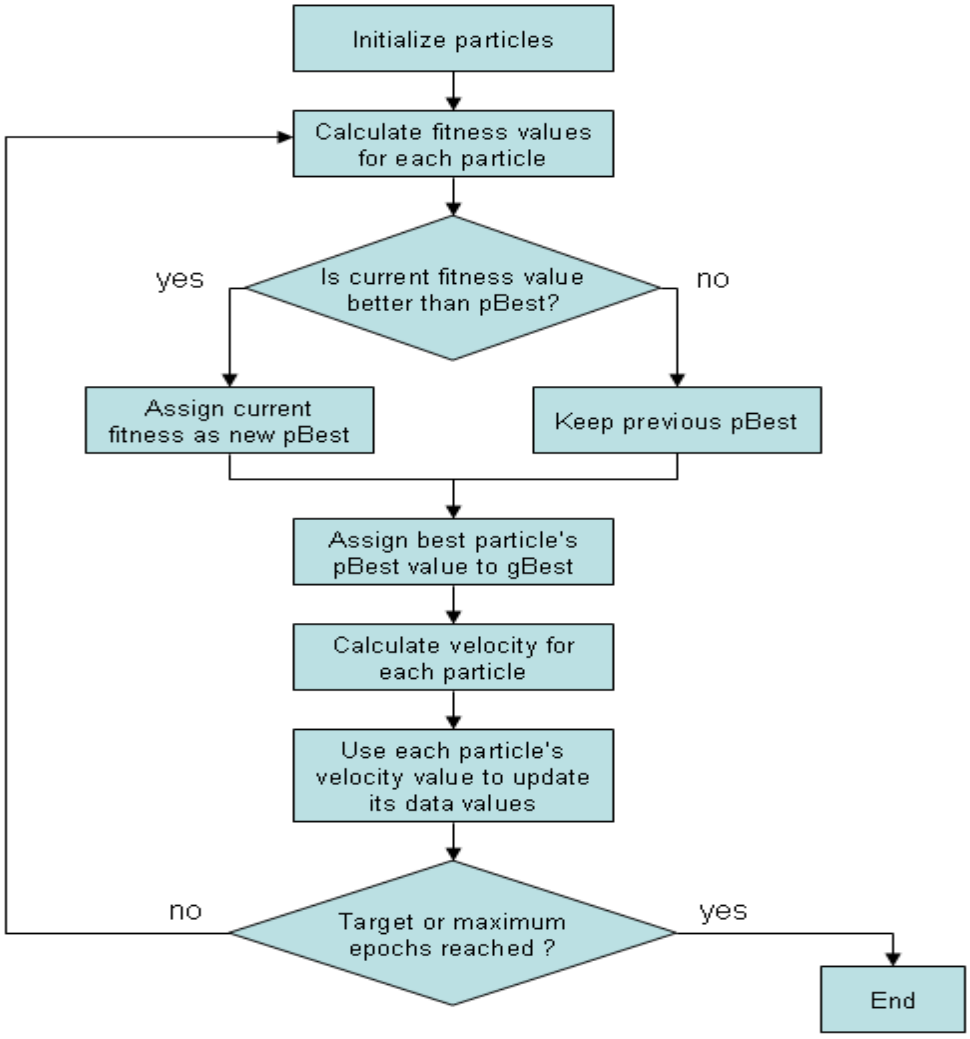

Fig.1 : Particle Swarm Optimization Control Flow Diagram

Ant Colony Optimization

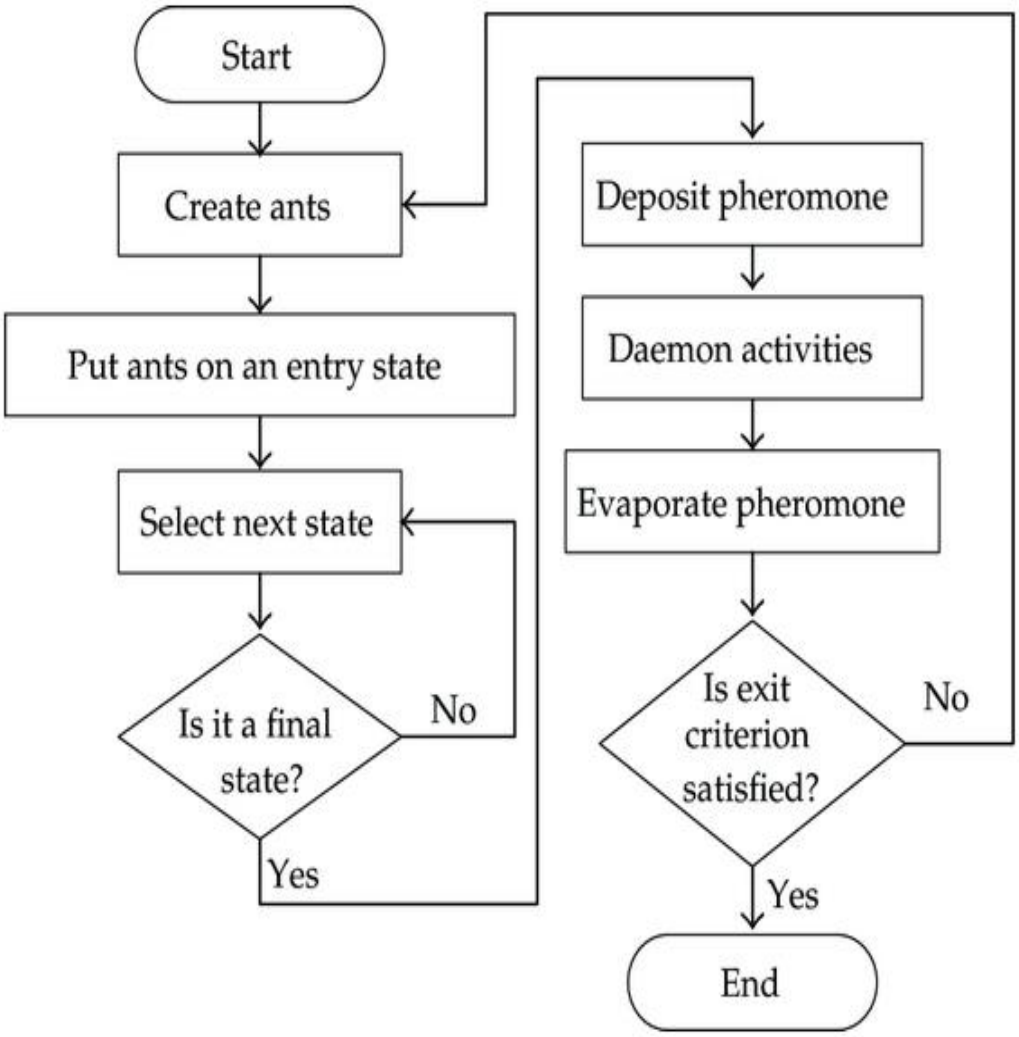

Fig 2: Ant Colony Optimization Control Flow Diagram

Chapter-4
Implementation details and issues

4.1 Implementation

4.1.1 Implementation details

TSP using Particle Swarm Optimization

TSP-PSO-1

Initialization: each of the particles gets a random solution and a random Swap Sequence, namely velocity.

TSPPSO-2

If the algorithms are ended, go to TSP-PSO-5.

TSP-PSO-3 .

For all the particles in position xid, calculating the next position Xid'

3-1 Calculating difference between Pid and Xid according to the method we have proposed above,

A=Pid-Xid, where A is a basic sequence.

3-2 Calculating B=Pgd-Xid, B is also a basic sequence

3-3 Calculating velocity Vid according to formula, and then we transform swap Sequence Vid to a Basic Swap Sequence.

3-4 Calculating new solution

Xid=Xid+Vid (2)

Formula (2) means that Swap Sequence vid, acts on solution Xid to get a new solution.

3-5 Updating Pid if tlie new solution is superior to Pid

TSP-PSO-4

Updating Pgd if there is new best solution ,which is Superior to Pgd. Goto TSP-PSO-2.

TSP-PS0-5

Sketch the global best solution.

TSP using Ant Colony Optimization

Moving of virtual ant depends on the amount of pheromone on the graph edges. The probability *pik* of transition of a virtual ant from the node *i* to the node *k* is given by formula below. We assume the existence of internal ant's memory.

$$p_i^k = \frac{\tau_i^\alpha + \eta_i^\beta}{\sum(\tau_{N^i}^\alpha + \eta_{N^i}^\beta)}$$

where

$\tau i\_$ - indicates the attractiveness of transition in the past

$\eta i\_$ - adds to transition attractiveness for ants,

Ni - set of nodes connected to point *i*, without the last visited point before *i*,

α, β - parameters found by simulation.

4.1.2 Algorithms

a) Particle Swarm Optimization

Pseudo Code

For each particle

{

Initialize particle

}

Do until maximum iterations or minimum error criteria

{

For each particle

{

Calculate Data fitness value

If the fitness value is better than pBest

{

Set pBest = current fitness value

}

If pBest is better than gBest

{

Set gBest = pBest

}

}

For each particle

{

Calculate particle Velocity

Use gBest and Velocity to update particle Data

}

The discrete PSO algorithm for TSP can be described as follows:

Step I :

Produce random particle and a random solution space called as velocity Vi.

Step 2.

Compute the next position for each particle in position Xi.

Step 3.

Compute the difference between next position and Xi, and repeat this until all positions are covered and the particle is at the position from where it has started, and keep adding the difference between each new position and the distance travelled until last position.

Mark this whole distance as Pbest and find the minimum Pbest from all the particles and mark it as Gbest.

Calculate a new velocity by the formula:

$$Vi = V\_MAX * (Gbest / Pbest)$$

Here Vi is the velocity of particle , V_MAX is a max velocity value .

Calculate a new solution :

$$Xi = Xi + Vi$$

Step 4.

Update Pbest and Gbest by repeating the procedure with new positions of particles

Step 5:

goto step 2 until termination criterion is satisfied.

Ant Colony Optimization

Initialize:

Set time:=0 {time is time counter}

For every edge (i, j) set an initial sij = c for trail density

and Dsij = 0.

Set s:=0 {s is travel step counter}

For k:=1 to l do

Place ant k on a city randomly. Place the city in visited k.

Place the group of the city in tabu k.

Repeat until s 6 m

Set s:=s + 1

For k:=1 to l do

Choose the next city to be visited according to the

probability pij

k given by Eq. (1).

Move the ant k to the selected city.

Insert the selected city in visited k.

Insert the group of selected city in tabu k.

For k:=1 to l do

Move the ant k from visited k(n) to visited k(1).

Compute the tour length Lk traveled by ant k.

Update the shortest tour found.

For every edge (i, j) do

For k:=1 to l do

Update the pheromone trail density sij according to Eqs.

(2)–(4).

time:=time + 1

If (time<TIME_MAX) then

Empty all visited k and tabu k

Goto Step 2.

Else

Print the shortest tour.

Stop

4.2) Risk Analysis and Mitigation Plan

Risk id	Classification	Description of Risk	Risk Area	Probability	Impact	RE(P*I)
1	Project Scope	Scope of project may not be clear	The whole project.	50%	High	0.75
2	Personell Related	Maybe some Personal Related discrepancies	The whole project	30%	Medium	0.51
3	Performance	The final Performance of project may not be upto the mark.	The outcome of project	60%	High	0.89
4	Testing Environment	The availability of a fine Environment may be diminished.	Testing of the project	20%	Low	0.20
5	External Input	The external inputs may not be upto the mark.	The testing of the project.	10%	Low	0.17
6	Hardware Components	The hardware components may not be functioning properly	The outcome of the project	30%	High	0.66
7	Open Source Code	The source code should be open	The availibily of project	5%	Low	0.05

Chapter-5
Testing

5.1) TESTING

Executing a program with the intent of finding errors is called testing. Testing is vital to the success of any system. Testing is done at different stages within the development phase. System testing makes a logical assumption that if all parts of the system are correct, the goals will be achieved successfully. Inadequate testing or no testing at all leads to errors that may come up after a long time when correction would be extremely implementation. The testing of the system was done on both artificial and live data. In order to test data test cases are developed.

Following are the various methods that are employed for testing:

5.1.1 Unit Testing

In unit testing the module is tested independently. It is done to test that the module does satisfy the functional specification. This is done to check syntax and logical errors in programs. At the time of preparation of technical specifications, unit test data was also prepared. The coding for that program was considered after verifying its output against this test data.

Following are the unit testing methods:

- In Conditional Testing, the logical conditions that are given in the module were checked to see whether they satisfy the functionality of the module. This is done by using the test data was prepared.
- In Loop Testing, different loops in the module like nested loops were tested using the data. Attempts to execute the loops to their maximum range are done.

5.1.2 Integration Testing

In Integration testing whole system was checked when all the individual modules were integrated together in order to test whether the system is performing as according to the requirements specified. Interface errors if any were corrected.

5.1.3 Functional Testing

This is done for each module/sub module of the system. Functional testing serves as a means of validating whether the functionality of the system confers the original user requirement i.e. does the module do what it was supposed to do? Separate schedules were made for functional testing. It involves preparation of test data, writing of test cases, testing for conformance to test cases and preparation of bugs' listing for nonconformities.

5.1.4 System Testing

System testing is done when the entire system has been fully integrated. The purpose of the system testing is to test how the different modules interact with each other and whether the entire system provides the functionality that was expected.

System testing consists of the following steps:

- **Program Testing**
- **System Testing**
- **System Documentation**
- **User Acceptance Testing**

5.2 Test Plan

The test-plan is basically a list of test cases that need to be run on the system. Some of the test cases can be run independently for some components (report generation from the database, for example, can be tested independently) and some of the test cases require the whole system to be ready for their execution. It is better to test each component as and when it is ready before integrating the components.

No.	Type of test	Will test be performed	Comments/explanations	Software component
1	Requirement testing	Yes	Checking minimum requirements to run the code.	Netbeans IDE

2	Unit testing	Yes	To check any type of errors or warnings in the code.	Netbeans IDE
3	Integration	Yes	To check if code produces any error when connected with dataset and if it is able to read values from dataset.	Netbeans IDE ,Microsoft excel
4	Performance	Yes	To check the overall performance of program limit of values it could handle.	Netbeans IDE
5	Load	Yes	To check if program hangs up when performing with large values.	Netbeans IDE
6	Security	No	There are no such security measures implemented on the working of program	Netbeans IDE

5.3 Component decomposition and type of testing required

S.no	List of various components that require testing	Type of testing required	Technique for test case
1	NetBeans IDE	Unit Testing	White box testing
2	Csv Library	Integration Testing	Black box testing
3	Dataset	Functional Testing	Black box testing
4	Input of values	Performance	Black box testing

Chapter-6
Findings & Conclusion

6.1 Findings

TSP is a good ground for testing optimization techniques, many researchers in various fields such as artificial intelligence, biology, mathematics, physics, and operations research devote themselves to trying to find the efficient methods for solving the TSP, such as genetic algorithms (GAs) , ant colony optimization (ACO) , simulated annealing (SA) , neural networks (NN) [, particle swarm optimization (PSO) , evolutionary algorithms (EA) [memetic computing , etc]. Besides, there are many practical applications of the TSP in the real world, such as data association, vehicle routing (with the additional constraints of vehicle's route, such as capacity's vehicles), data transmission in computer networks, job scheduling, DNA sequencing, drilling of printed circuits boards, clustering of data arrays, image processing and pattern recognition, analysis of the structure of crystals, transportation and logistics.

Swarm intelligence is an important research topic based on the collective behavior of decentralized and self-organized systems in computational intelligence. It consists of a population which simulates the animals' behavior in the real world. Now there are many swarm intelligence optimization algorithms, such as genetic algorithms, particle swarm optimization, ant colony optimization, bee colony algorithm, differential evolution, fish-warm algorithm, etc. Due to the simple concept, easy implementation and quick convergence, PSO has gained much attention and been successfully applied in a variety of fields mainly for optimization problems.

6.2 CONCLUSION

This project tried to cover the state-of-the-art studies about Swarm Intelligence algorithms and their application to solve the classical NP-Hard travelling salesman problem. It covers recent thesis reports and introduces the latest developed protocols based on PSO and ACO.

It has been a great pleasure to study these papers. At first I worked clustering techniques which gave me basic idea of clustering methods and then I worked on optimization techniques which helped me to extend them to use as a approach to solve travelling salesman problem. Thus, the applications of Swarm Intelligence optimization techniques (Particle Swarm Optimization and Ant Colony Optimization) on Travelling Salesman Problem along are easier to implement and understand since the main ideas behind them have been inspired by nature.

I hope this report gave you a good understanding of the Optimization Techniques and their implementation on Travelling salesman problem and therefore, entire satisfaction.

6.3 FUTURE WORK

Swarm Intelligence is definitely a big player when it comes to AI. However as good as it looks there are still far more important things to tackle in AI, most important of all I would say is to not only solve but define what human consciousness is: and thus imitate our own superior form of intelligence to machines.

Today, AI has reached very far compared to how it started out at least 60 years ago, Machine Learning and PGM's are pushing the frontier in intelligent decision modelling as well as Deep Learning. Yet as more papers keep on getting published and the ROC curves keep getting better, with higher precision and recall rates, we still have a vague idea of what intelligence is beyond the capacity of taking a decision.

Swarm intelligence is a topic under intense research for a long time. Day by day new and better intelligence techniques are being introduced and the overall scenario of AI is getting better. I selected this topic because the domain machine learning got a lot to explore and so I wanted to explore some of its areas like clustering and optimization. I enjoyed working on this project and learned a lot of new things which I will use in some of my future projects. I learned that there is no limit to optimization and so it’s a good area to explore extensively with number of various applications.

REFERENCES

Reports:

[1] E. Bonabeau, M. Dorigo, and G. Theraulaz. Swarm intelligence:from natural to artificial *intelligence*. Oxford University Press, 1999. ISBN 0-19-513158-4.

[2] Reinelt, G.: The Traveling Salesman. Springer, Berlin (1994)

[3] Stadler, P.F., Schnabl, W.: The landscape of the traveling salesman problem. Phys. Lett. A 161, 337–344 ,1992.

[4] Solve Traveling Salesman Problem Using Particle Swarm Optimization Algorithm ,2012, Xuesong Yan, Can Zhang, Wenjing Luo, Wei Li, Wei Chen and Hanmin Liu , School of Computer Science, China University of Geosciences Wuhan, Hubei 430074, China ,Wuhan Institute of Ship Building Technology Wuhan, Hubei 430050, China

[5] Particle swarm optimization for traveling salesman problem,2007, kang-ping wang, lan huang, chun-guang zhou, wei pang , College of Computer Science and Technology, Jilin University, Changchun 130012, China E-MAIL: cgzhou@mail jlu.edu.cn

Handbooks :

[1] M. Dorigo and G. Di Caro. The ant colony optimization meta-heuristic. In D. Corne, M. Dorigo, and F. Glover, editors, New Ideas in Optimization, pages 11 32. McGraw- Hill, London, 1999.

[2] Particle Swarm Optimization Book, by Veysel Gazi ,Kevin M. Passino ,2011

[3] Ant Colony Optimization Book, by Marco Dorigo and Thomas Stützle , 2002

APPENDIX

A) Project plan as Gantt Chart:-

TASK	TIME						
	Week 1 27 march	Week 2 3 april	Week 3 10 april	Week 4 17 april	Week 5 24 april	Week 6 1 may	Week 7 8 may
Basic of Travelling salesman problem	x						
Basic of PSO		X					
Research paper study			x				
Study related to implemen Tation with java				x			
Intial implementat Ion					x		
With Input from file					x		
Complete implement Ation						x	
Test							X

RESUME

GAURAV CHAUHAN

Mobile No. : 9711876016

Email Id: gaurav.16.chauhan@gmail.com

CAREER OBJECTIVE

To learn at every stage of life and always remain open for knowledge no matter where it came from. And to touch the zenith of success by the implication of that knowledge.

SUMMER INTERNSHIP/INDUSTRIAL TRAINING
Triotech Solutions PVT. LTD.

Duration: 3 July, 2016 to 02 September, 2016

Work: Web Development **(**Front End and Back End)

Language: HTML, CSS, JAVASCRIPT,PHP

Aptech Computer Education

Duration: 1 July, 2015 to 15 August, 2015

Project: Training for java ,java advance language

Language : IDE Used – NetBeans , Database – MySQL, Programming Java(J2SE)

NIIT
Duration: 5 July, 2014 to 20 August, 2014

Project: Training for C++ language

Language: C++

PROJECTS UNDERTAKEN

PRASEDIUM- ANDROID APPLICATION

Language used: JAVA

DATABASE: SQL

In Today's world mobile phones has become an indispensable part of a person's life. A life cannot be imagined without a smartphone, but with this need of smartphone the thought of fear of losing it is also there with the owner of the phone. Today theft of the smartphone is very easy and very convenient for the thieves because one in not able to find it back due to lack of support from police and some technical problems.

The objective of our project is to provide a fully featured anti-theft system with a fully functional system for getting the device back with little help of police. The application will run on android platform and will be helpful to the devices.

BROWSE-ME WEBSITE

Duration: 62 days

Language used: HTML, CSS,JAVASCRIPT,PHP

Database: MySQL

This project addresses the study of the emerging sector of website development, and its core features that are used by most of the websites running today. The main goal is to develop an accessible and user friendly platform for users who wants to connect with various websites of different categories from a single platform with choices from similar other websites of the same category in case the user wants to try other similar website that he is using and for users who wants to learn how to use various websites like social media websites , entertainment websites , coding websites etc and also to keep the users updated with latest news of any event happened regarding any website like rolling out of new features or acquiring of a website by some other big firm and all such news. The user can browse through different categories from homepage only, by shortcut links provided on homepage, but if they want to use other features, like browse through categories , and learning to use any website , they have to signup/login through a very simple procedure .

SKILL SET AND AREAS OF INTEREST

Known Programming Languages – C, C++, JAVA (J2EE)

Html, CSS, PHP,JavaScript, Android.

Areas of Interest:

Front End Development Logic Building

Data Analysis

Web Application Development Web Designing

Back End Development

PERSONAL DETAILS

Date of Birth – 19 September, 1995

Contact No. – 9711876016

Nationality – Indian

Permanent Address – H.No-124,Cross Road,Sant Nagar, Delhi – 110084,

DECLARATION

I hereby declare that the details provided by me in this resume are correct to the best of my knowledge.