Table of Contents

[CHAPTER 1 INTRODUCTION 9](#_Toc408316285)

[1.1 Overview 9](#_Toc408316287)

[1.2 Particle Swarm Optimization (PSO) 10](#_Toc408316288)

[1.3 Genetic Algorithm 11](#_Toc408316289)

[1.4 Motivation 13](#_Toc408316290)

[1.5 Problem Statement and Objectives 14](#_Toc408316291)

[1.6 Organization of the Report 14](#_Toc408316292)

[CHAPTER 2 LITERATURE SURVEY 15](#_Toc408316293)-26

[2.1 Travelling Salesman Problem 15](#_Toc408316295)

[2.1.1 Hamiltonian Path 17](#_Toc408316296)

[2.2 Genetic Algorithm 17](#_Toc408316297)

[2.2.1 Genetic Coding 18](#_Toc408316298)

[2.2.2 Fitness Function 19](#_Toc408316299)

[2.2.3 Reproduction Operator 19](#_Toc408316300)

[2.2.4 Sequential Constructive Crossover Operator (SCX) 19](#_Toc408316301)

[2.2.5 Offspring by Two Other Crossover Operators 21](#_Toc408316302)

[2.2.6 Survivor Selection 22](#_Toc408316303)

[2.2.7 Mutation Operator 23](#_Toc408316304)

[2.2.8 Control Parameters 23](#_Toc408316305)

[2.3 Particle Swarm Optimization (PSO) for Network Optimization 23](#_Toc408316306)

[2.3.1 Classical PSO 24](#_Toc408316307)

[2.3.2 Particle Swarm Optimization 25](#_Toc408316308)

[2.3.3 Basic PSO Algorithm 26](#_Toc408316309)

[CHAPTER 3 PROPOSED METHOD 27](#_Toc408316310)-44

[3.1 Travelling Salesman Problem Using PSO 27](#_Toc408316312)

[3.1.1 How it works? 28](#_Toc408316313)

[3.1.2 Procedure 29](#_Toc408316314)

[3.1.3 Program Module for PSO applied on TSP 34](#_Toc408316315)

[3.2 Genetic Algorithm for Solving TSP 39](#_Toc408316316)

[3.2.1 Implementation of Proposed Algorithm 40](#_Toc408316317)

[3.2.2 Initial Population 42](#_Toc408316318)

[3.2.3 Fitness Value 43](#_Toc408316319)

[3.2.4 Selection 43](#_Toc408316320)

[3.2.5 Roulette Wheel Selection Method 44](#_Toc408316321)

[3.2.6 Crossover 44](#_Toc408316322)

[3.2.7 Mutation 44](#_Toc408316323)

[3.2.8 Termination and Result 44](#_Toc408316324)

[CHAPTER 4 EXPERIMENTAL RESULT AND ANALYSIS 45](#_Toc408316325)-48

[4.1 Result Obtained after Application of PSO on TSP 45](#_Toc408316327)

[4.2 Result Obtained after Application of GA on TSP 47](#_Toc408316328)

[4.2.1 Output on Running Genetic Algorithm on Travelling Salesman Problem 48](#_Toc408316329)

[CHAPTER 5 CONCLUSION AND FUTURE WORK 49](#_Toc408316330)

[5.1 Conclusion 49](#_Toc408316332)

[5.2 Future Work 49](#_Toc408316333)

[REFERENCES 50](#_Toc408316334)

**CHAPTER 1**

# INTRODUCTION

## Overview

With the increasing progress of both computer software and hardware, a large amount of data is generated and collected from different sources daily. Data collected contains useful information which is hidden so one can extract useful information from this data by discovering patterns in the data. However, the major problem for obtaining useful knowledge from data lies in the data itself. Therefore, Data mining plays a important role in extracting knowledge and useful patterns from the data.

Recently, data management and processing for wireless sensor networks (WSNs) has become a topic for research in several fields of computer science, such as distributed system, the database system, and the data mining [1]. We have proposed Clustering a data mining approach for making efficient clusters in wireless Sensor Network. Different sensor receives data and form clusters using k-prototype clustering algorithm discussed in chapter 2 and transmit this data to the base stations.

Traditional data mining techniques are not directly applicable to WSNs due to the nature of sensor data, their special characteristics and limitations of the WSNs. Advances in wireless communication led to the development of low-power sensors and the deployment of large-scale sensor networks.

The rapid development of information technology has leaded to the generation of data at an exponential rate in a wide range of areas, such as business, society, engineering and healthcare. The emergence of large-scale data requires new techniques that can extract useful information and knowledge from them. Data mining is a suitable technique to satisfy this need [[1]](http://www.sciencedirect.com/science/article/pii/S0925231213004773" \l "bib1), [[2]](http://www.sciencedirect.com/science/article/pii/S0925231213004773" \l "bib2) and [[3]](http://www.sciencedirect.com/science/article/pii/S0925231213004773" \l "bib3).

In data mining, clustering is an important technique. The clustering algorithms are widely used in image processing, customer segmentation, gene expression analysis [[4]](http://www.sciencedirect.com/science/article/pii/S0925231213004773" \l "bib4), and text documents analysis [[5]](http://www.sciencedirect.com/science/article/pii/S0925231213004773" \l "bib5) etc. The aim of clustering is to divide a set of data objects into clusters such that data objects in the same cluster are more similar to each other than those in other clusters [[6]](http://www.sciencedirect.com/science/article/pii/S0925231213004773" \l "bib6), [[7]](http://www.sciencedirect.com/science/article/pii/S0925231213004773" \l "bib7), [[8]](http://www.sciencedirect.com/science/article/pii/S0925231213004773" \l "bib8) and [[9]](http://www.sciencedirect.com/science/article/pii/S0925231213004773" \l "bib9). In real world, data sets usually contain both numeric and categorical attributes [[10]](http://www.sciencedirect.com/science/article/pii/S0925231213004773" \l "bib10) and [[11]](http://www.sciencedirect.com/science/article/pii/S0925231213004773" \l "bib11). However, most existing clustering algorithms assume all attributes are either numeric or categorical, examples of which include the k-means [[12]](http://www.sciencedirect.com/science/article/pii/S0925231213004773" \l "bib12), k-modes [[13]](http://www.sciencedirect.com/science/article/pii/S0925231213004773" \l "bib13), fuzzy k-modes [[14]](http://www.sciencedirect.com/science/article/pii/S0925231213004773" \l "bib14), TGCA [[15]](http://www.sciencedirect.com/science/article/pii/S0925231213004773" \l "bib15), COOLCAT [[16]](http://www.sciencedirect.com/science/article/pii/S0925231213004773" \l "bib16), and G-ANMI algorithms [[17]](http://www.sciencedirect.com/science/article/pii/S0925231213004773" \l "bib17). When mixed data are encountered, most of them usually exploit transformation approaches to convert one type of the attributes to the other and then apply traditional single-type clustering algorithms [[3]](http://www.sciencedirect.com/science/article/pii/S0925231213004773" \l "bib3). However, in most cases, transformation scheme may result in loss of information, leading to undesired clustering outcomes[[11]](http://www.sciencedirect.com/science/article/pii/S0925231213004773" \l "bib11).

Clustering is an established data mining technique for grouping data based on similarity. This technique is useful for many applications such as wireless sensor networks (WSNs). WSNs consist usually of sensor nodes which are energy, processing and storage limited devices. Sensor nodes are usually distributed over a certain area covered by the WSN to observe phenomena (light, temperature, humidity) by collecting data. Physical clustering of sensor nodes aims at grouping together sensor nodes that are sensing correlated data and selecting one of them as a representative, while turning others off [2]. This minimizes the energy consumption and thus extends the lifetime of the nodes.

Data mining in sensor networks is the process of extracting application-oriented models and patterns with acceptable accuracy from a continuous, rapid, and possibly non ended flow of data streams from sensor networks. In this case, whole data cannot be stored and must be processed immediately. Data mining algorithm has to be sufficiently fast to process high-speed arriving data. The conventional data mining algorithms are meant to handle the static data and use the multistep techniques and multi scan mining algorithms for analyzing static data-sets. Therefore,

Conventional data mining techniques are not suitable for handling the massive quantity, high dimensionality, and distributed nature of the data generated by the WSNs

The k-means algorithm is well known for its efficiency in clustering large data sets. However, working only on numeric values prohibits it from being used to cluster real world data containing categorical values. In our project we have used k-prototype algorithm which extend the k-means algorithm to categorical domains and domains with mixed numeric and categorical values. The k -modes algorithm uses a simple matching dissimilarity measure to deal with categorical objects, replaces the means of clusters with modes, and uses a frequency-based method to update modes in the clustering process to minimize the clustering cost function [3]. With these extensions the k–modes algorithm enables the clustering of categorical data in a fashion similar to k-means. The k-prototypes algorithm, through the definition of a combined dissimilarity measure, further integrates the k-means and k -modes algorithms to allow for clustering objects described by mixed numeric and categorical attributes

## Wireless Sensor Network

A wireless sensor network (WSN**)** (sometimes called a wireless sensor and actor network(WSAN)are spatially distributed autonomous sensors to monitor physical or environmental conditions, such as temperature, sound, pressure, etc. and to cooperatively pass their data through the network to a main location. The more modern networks are bi-directional, also enabling control of sensor activity. The development of wireless sensor networks was motivated by military applications such as battlefield surveillance; today such networks are used in many industrial and consumer applications, such as industrial process monitoring and control, machine health monitoring, and so on.

* 1. Clustering

Clustering can be considered the most important unsupervised learning problem; so, as every other problem of this kind, it deals with finding a structure in a collection of unlabeled data.  
A loose definition of clustering could be “the process of organizing objects into groups whose members are similar in some way”.  
A cluster is therefore a collection of objects which are “similar” between them and are “dissimilar” to the objects belonging to other clusters.

the goal of clustering is to determine the intrinsic grouping in a set of unlabeled data. But how to decide what constitutes a good clustering? It can be shown that there is no absolute “best” criterion which would be independent of the final aim of the clustering. Consequently, it is the user which must supply this criterion, in such a way that the result of the clustering will suit their needs.  
For instance, we could be interested in finding representatives for homogeneous groups (data reduction), in finding “natural clusters” and describe their unknown properties (“natural” data types), in finding useful and suitable groupings (“useful” data classes) or in finding unusual data objects (outlier detection).

The primary use of clustering algorithms is to discover the grouping structures inherent in data. For this purpose an assumption is first made that a certain structure may exist in a given data set and then a clustering algorithm is used to verify the assumption and recover the structure. Jain and Dubes (1988) discussed three types of criteria used to evaluate the performance of clustering methods in discovering the inherent data structures.

* 1. k-means clustering

K-means is one of the simplest unsupervised learning algorithms that solve the well known clustering problem. The procedure follows a simple and easy way to classify a given data set through a certain number of clusters (assume k clusters) fixed a priori. The main idea is to define k centroids, one for each cluster. These centroids should be placed in a cunning way because of different location causes different result. So, the better choice is to place them as much as possible far away from each other. The next step is to take each point belonging to a given data set and associate it to the nearest centroid. When no point is pending, the first step is completed and an early grouping is done. At this point we need to re-calculate k new centroids as barycenters of the clusters resulting from the previous step. After we have these k new centroids, a new binding has to be done between the same data set points and the nearest new centroid. A loop has been generated. As a result of this loop we may notice that the k centroids change their location step by step until no more changes are done. In other words centroids do not move any more.  
Finally, this algorithm aims at minimizing an objective function, in this case a squared error function. The objective function

1.4 k-mode

The cause that the k-means algorithm cannot cluster categorical objects is its dissimilarity measure These barriers can be removed by making the following modifications to the k-means algorithm:

1. using a simple matching dissimilarity measure for categorical objects,

2. replacing means of clusters by modes, and

3. using a frequency-based method to find the modes

Like the k-means algorithm the k-modes algorithm also produces locally optimal solutions that are dependent on the initial modes and the order of objects in the data set

The k-modes algorithm extends the k-means algorithm for clustering categorical data by using a simple matching dissimilarity measure for categorical objects, modes instead of means for clusters and a frequency based method to update the modes using the k-means method to minimize the cost of clustering cost function [http://arxiv.org/ftp/cs/papers/0603/0603120.pdf].

1.5 k-prototype

In our project we used an k-prototypes algorithm for clustering mixed data. In our method, we first named the distribution centroid to represent the prototype of categorical attributes in a t cluster [http://www.sciencedirect.com/science/article/pii/S0925231213004773]. Then we integrate the mean with distribution centroid to represent the prototype of a cluster with mixed attributes, and propose a new dissimilarity measure, which takes account of the significance of each attribute, to evaluate the dissimilarity between data objects and prototypes.

The algorithm is used to present an effective representation for the categorical attribute part in a mixed prototype since the mean is good enough for the numeric attribute part, and on the other hand to consider the significance of different attributes towards the clustering process.

It is straightforward to integrate the k-means and k-modes algorithms into the k-prototypes algorithm that is used to cluster the mixed-type objects. The k-prototypes algorithm is practically more useful because frequently encountered objects in real world databases are mixed-type objects.

## 1.4 Motivation

As the data collected from different sources consists of both categorical and numerical values

so, the conventional approaches like k-means clustering does not fit well for data containing both kind of attributes. So we have used new approaches which are extensions for existing clustering algorithms. k-modes algorithm and k-prototype algorithms are used for data containing both kind of attributes categorical and numerical. These new approaches uses data mining for finding similarity measures between the attribute values for different data objects.

The main objective of using these algorithm is to minimize the cost function discussed in later section.

As the wireless sensor network collects incoming data like temperature, humidity and rainfall on the sensors and sends the collected data to the base stations but the collected data varies a lot.

So efficient clusters are formed so that a cluster contains homogeneous data. Therefore data objects with attribute values which have the most similiarity measure are put in the same cluster.

## 1.5 Problem Statement and Objectives

The main objective of this proposed work is to apply Particle Swarm Optimization (PSO) and Gen't netic Algorithm (GA) to solve Travelling Salesman Problem and compare the results obtained by both of them which can be useful for finding a shortest path in a network in less time.

The work done is basically divided into two parts:

1. Firstly, we have applied PSO on the TSP and recorded and compared the result with different particle count i.e. no of routes we have considered and maximum velocity i.e. maximum change allowed for each particle.
2. Secondly, we have applied GA on the TSP and recorded the results by generating new offspring from the existing population of chromosomes by using crossover and mutation techniques.

Finally, we compared the results obtained by the application of PSO and GA on the TSP.

## 1.6 Organization of the Report

The report is divided into 5 chapters. Explanation about each chapter is given below.

**Introduction chapter** gives an overview about the routing problems and a brief introduction about TSP, PSO and GA.

**Literature Survey** contains introduction about the basic idea about Particle Swarm Optimization, Genetic Algorithm, and Travelling Salesman Problem. It also explains how PSO and GA can be applied on TSP

**Proposed Method** chaptercontains the methodology used for the application of PSO and GA for solving TSP

**Experimental Result and Analysis** chaptercontains the result obtained on application of PSO and GA separately. The results obtained are represented using two dimensional spaces as a coordinates for cities. The results obtained for PSO and GA are then compared.

**Conclusion and Future Work** chapter gives overall concluded results which shows that PSO works faster than GA.

# CHAPTER 2

# LITERATURE SURVEY

## 2.1 Data mining

Data mining, or knowledge discovery, is the computer-assisted process of digging through and analyzing enormous sets of data and then extracting the meaning of the data. Data mining tools predict behaviors and future trends, allowing businesses to make proactive, knowledge-driven decisions. Data mining tools can answer business questions that traditionally were too time consuming to resolve.

Generally, data mining (sometimes called data or knowledge discovery) is the process of analyzing data from different perspectives and summarizing it into useful information - information that can be used to increase revenue, cuts costs, or both. Data mining software is one of a number of analytical tools for analyzing data. It allows users to analyze data from many different dimensions or angles, categorize it, and summarize the relationships identified. Technically, data mining is the process of finding correlations or patterns among dozens of fields in large relational databases.  
 They scour databases for hidden patterns, finding predictive information that experts may miss because it lies outside their expectations.

Data mining derives its name from the similarities between searching for valuable information in a large database and mining a mountain for a vein of valuable ore. Both processes require either sifting through an immense amount of material, or intelligently probing it to find where the value resides.

2.2 Data Mining Techniques

There are several major data mining  techniques  have been developing and using in data mining projects recently including association,classification, clustering, prediction, sequential patterns and decision tree.

2.2.1 data preprocessing

Raw data is highly susceptible to noise, missing values, and inconsistency. The quality of

data affects the data mining results. In order to help improve the quality of the data and,

consequently, of the mining results raw data is preprocessed so as to improve the

efficiency and ease of the mining process. Data preprocessing is one of the most critical

steps in a data mining process which deals with the preparation and transformation of the

initial dataset.

Data preprocessing methods are divided Into following categories:

1. data cleaning

2. data integration

3. data transformation

4. data reduction

2.2.1.1 Data Cleaning

Data cleaning routines work to “clean” the data by filling in missing values, smooth-

ing noisy data, identifying or removing outliers, and resolving inconsistencies. If users

believe the data are dirty, they are unlikely to trust the results of any data mining that

has been applied to it. Furthermore, dirty data can cause confusion for the mining pro-

cedure, resulting in unreliable output. Although most mining routines have some pro-

cedures for dealing with incomplete or noisy data, they are not always robust. Instead,

they may concentrate on avoiding overfitting the data to the function being modeled.

2.2.1.2 Data Integration

It is likely that your data analysis task will involve data integration, which combines data from multiple sources into a coherent data store, as in data warehousing. These sources may include multiple databases, data cubes, or flat files. There are a number of issues to consider during data integration. Schema integration can be tricky. How can

like real world entities from multiple data sources be 'matched up'? This is referred to as the entity

identification problem. For example, how can the data analyst or the computer be sure that customer id in one database, and customer\_number in another refer to the same entity? Databases and data warehouses typically have metadata that is, data about the data. Such metadata can be used to help avoid errors in schema integration. Redundancy is another important issue. An attribute may be redundant if it can be derived from another table, such as annual revenue. Inconsistencies in attribute or dimension naming can also cause redundancies in the resulting data set.

2.2.1.3 Data Transformation

In data transformation, the data are transformed or consolidated into forms appropriate for mining. Data transformation can involve the following:

1. Normalization, where the attribute data are scaled so as to fall within a small specified range, such as – 1.0 to 1.0, or 0 to 1.0.

2. Smoothing works to remove the noise from data. Such techniques include binning, clustering, and regression.

3. Aggregation, where summary or aggregation operations are applied to the data. For example, the daily sales data may be aggregated so as to compute monthly and annual total amounts. This step is typically used in constructing a data cube for analysis of the data at multiple granularities.

4. Generalization of the data, where low level or 'primitive' (raw) data are replaced by higher level concepts through the use of concept hierarchies. For example, categorical attributes, like street, can be generalized to higher level concepts, like city or county. Similarly, values for numeric attributes, like age, may be mapped to higher level concepts, like young, middle-aged, and senior.

2.2.1.4 Data Reduction

Complex data analysis and mining on huge amounts of data may take a very long time, making such analysis impractical or infeasible. Data reduction techniques have been helpful in analyzing reduced representation of the dataset without compromising the integrity of the original data and yet producing the quality knowledge. The concept of data reduction is commonly understood as either reducing the volume or reducing the dimensions (number of attributes). There are a number of methods that have facilitated in analyzing a reduced volume or dimension of data and yet yield useful knowledge. Certain partition based methods work on partition of data tuples. That is, mining on the reduced data set should be more efficient yet produce the same (or almost the same) analytical results.

Strategies for data reduction include the following.

1. Data cube aggregation, where aggregation operations are applied to the data in the construction of a data cube.

2. Dimension reduction, where irrelevant, weakly relevant, or redundant attributes or dimensions may be detected and removed.

3. Data compression, where encoding mechanisms are used to reduce the data set size. The methods used for data compression are wavelet transform and Principal Component Analysis.

4. Numerosity reduction, where the data are replaced or estimated by alternative, smaller data representations such as parametric models (which need store only the model parameters instead of the actual data e.g. regression and log-linear models), or nonparametric methods such as clustering, sampling, and the use of histograms.

5. Discretization and concept hierarchy generation, where raw data values for attributes are replaced by ranges or higher conceptual levels. Concept hierarchies allow the mining of data at multiple levels of abstraction, and are a powerful tool for data mining.

2.2.1 Association

Association is one of the best known data mining technique. In association, a pattern is discovered based on a relationship between items in the same transaction. That’s is the reason why association technique is also known as relation technique. The association technique is used in market basket analysis to identify a set of products that customers frequently purchase together.

Retailers are using association technique to research customer’s buying habits. Based on historical sale data, retailers might find out that customers always buy crisps when they buy beers, and therefore they can put beers and crisps next to each other to save time for customer and increase sales.

2.2.2 Classification

Classification is a classic data mining technique based on machine learning. Basically classification is used to classify each item in a set of data into one of predefined set of classes or groups. Classification method makes use of mathematical techniques such as decision trees, linear programming, neural network and statistics. In classification, we develop the software that can learn how to classify the data items into groups.

Classification is a task in Data mining. Data mining, as indicated before, is a machine learning discipline, and is inspired by pattern recognitions, which is a branch of science, of which one of its goals is to classify objects into a number of categories referred to as classes. Objects refer to compact data units specific to a particular problem, which is in general, known as patterns.

Classification prediction encompasses two levels: classifier construction and the usage of the classifier constructed. The former is concerned with the building of a classification model by describing a set of predetermined classes from a training set as a result of learning from that dataset. Each sample in the training set is assumed to belong to a predefined class, as determined by the class attribute label. The model is represented as classification rules, decision trees , or mathematical formula. The later involves the use of a classifier built to predict or classify unknown objects based on the patterns observed in the training set.

The entire process begins with collection of evidence acquired from various data sources or warehouses. In the ideal situation, the data should be of low-dimensionality, independent and

discriminative so that its values are very similar to characteristics in the same class but very different in features from different classes. Raw data hardly satisfies these conditions and therefore a set of procedures called feature generation, extraction and selection is required to provide a relevant input for classification system.

For example, we can apply classification in application that “given all records of employees who left the company, predict who will probably leave the company in a future period.” In this case, we divide the records of employees into two groups that named “leave” and “stay”. And then we can ask our data mining software to classify the employees into separate groups.

2.2.3 Clustering

Clustering is a data mining technique that makes meaningful or useful cluster of objects which have similar characteristics using automatic technique. The clustering technique defines the classes and puts objects in each class, while in the classification techniques, objects are assigned into predefined classes [ http://www.zentut.com/data-mining/data-mining-techniques/].

Clustering is a division of data into groups of similar objects. Representing the data by fewer clusters necessarily loses certain fine details, but achieves simplification. It models data by its clusters. Data modeling puts clustering in a historical perspective rooted in mathematics, statics, and numerical analysis. From a machine learning perspective clusters correspond to hidden patterns, the search for clusters is unsupervised learning, and the resulting system represents a data mining applications such as scientific data exploration, information retrieval and text mining, spatial database applications, web analysis, CRM, marketing, medical diagnostics, computational biology, and many others.

To make the concept clearer, we can take book management in library as an example. In a library, there is a wide range of books in various topics available. The challenge is how to keep those books in a way that readers can take several books in a particular topic without hassle. By using clustering technique, we can keep books that have some kinds of similarities in one cluster or one shelf and label it with a meaningful name. If readers want to grab books in that topic, they would only have to go to that shelf instead of looking for entire library.

2.2.4 Prediction

The prediction, as it name implied, is one of a data mining techniques that discovers relationship between independent variables and relationship between dependent and independent variables. For instance, the prediction analysis technique can be used in sale to predict profit for the future if we consider sale is an independent variable, profit could be a dependent variable. Then based on the historical sale and profit data, we can draw a fitted regression curve that is used for profit prediction.

2.2.5 Sequential Pattern

Sequential patterns analysis is one of data mining technique that seeks to discover or identify similar patterns, regular events or trends in transaction data over a business period.

In sales, with historical transaction data, businesses can identify a set of items that customers buy together a different times in a year. Then businesses can use this information to recommend customers buy it with better deals based on their purchasing frequency in the past.

2.2.6 Decision Trees

Decision tree is one of the most used data mining techniques because its model is easy to understand for users. In decision tree technique, the root of the decision tree is a simple question or condition that has multiple answers. Each answer then leads to a set of questions or conditions that help us determine the data so that we can make the final decision based on it

2.2.7 Clustering Techniques

2.2.7.1 k-means Clustering

The k-means algorithm (MacQueen, 1967; Anderberg, 1973), one of the mostly used clustering algorithms, is classified as a partitional or nonhierarchical clustering method (Jain and Dubes, 1988). Given a set of numeric objects X and an integer number k (≤n), the k-means algorithm searches for a partition of X into k clusters that minimises the within groups sum of squared errors (WGSS). This process is often formulated as the following mathematical program problem P [<http://www.cs.ust.hk/~qyang/Teaching/537/Papers/huang98extensions.pdf>]

Minimise P(W, Q) = X k l=1 Xn i=1 wi,l d(Xi, Ql) (1)

subject to X k l=1 wi,l = 1, 1 ≤ i ≤ n wi,l ∈ {0, 1}, 1 ≤ i ≤ n, 1 ≤ l ≤ k (2)

where W is an n ×k partition matrix, Q = {Q1, Q2,..., Qk } is a set of objects in the same object domain, and d(·, ·) is the squared Euclidean distance between two objects.

Problem P can be solved by iteratively solving the following two problems:

1. Problem P1: Fix Q = Qˆ and solve the reduced problem P(W, Qˆ ).

2. Problem P2: Fix W = Wˆ and solve the reduced problem P(Wˆ , Q).

The k-means algorithm has the following important properties:

1. It is efficient in processing large data sets.

2. It often terminates at a local optimum (MacQueen, 1967; Selim and Ismail, 1984).

3. It works only on numeric values.

4. The clusters have convex shapes

2.2.7.2 k-mode Clustering

In principle the formulation of problem P in Section 3 is also valid for categorical and mixedtype objects. The cause that the k-means algorithm cannot cluster categorical objects is its dissimilarity measure and the method used to solve problem P2. These barriers can be removed by making the following modifications to the k-means algorithm:

1. using a simple matching dissimilarity measure for categorical objects,

2. replacing means of clusters by modes, and

3. using a frequency-based method to find the modes to solve problem P2

2.2.7.3.1 Disimiliarity Measure

Let X, Y be two categorical objects described by m categorical attributes. The dissimilarity measure between X and Y can be defined by the total mismatches of the corresponding attribute categories of the two objects. The smaller the number of mismatches is, the more similar the two objects. This measure is often referred to as simple matching (Kaufman and Rousseeuw, 1990). Formally,

Eq 5

Eq 6 ([<http://www.cs.ust.hk/~qyang/Teaching/537/Papers/huang98extensions.pdf>])

2.2.7. Mode of a set

Let X be a set of categorical objects described by categorical attributes, A1, A2,..., Am. Definition 1. A mode of X = {X1, X2,..., Xn} is a vector Q = [q1, q2,..., qm] that minimises

Eq 7

2.2.7. k-mode

When (5) is used as the dissimilarity measure for categorical objects, the cost function (1) becomes

Eq (8)

where wi,l ∈ W and Ql = [ql,1, ql,2,..., ql,m] ∈ Q. To minimise the cost function the basic k-means algorithm can be modified by using the simple matching dissimilarity measure to solve P1, using modes for clusters instead of means and selecting modes according to Theorem 1 to solve P2. In the basic algorithm we need to calculate the total cost P against the whole data set each time when a new Q or W is obtained. To make the computation more efficient we use the following algorithm instead in practice.

1. Select k initial modes, one for each cluster.

2. Allocate an object to the cluster whose mode is the nearest to it according to (5). Update the mode of the cluster after each allocation according to Theorem 1.

3. After all objects have been allocated to clusters, retest the dissimilarity of objects against the current modes. If an object is found such that its nearest mode belongs to another cluster rather than its current one, reallocate the object to that cluster and update the modes of both clusters.

4. Repeat 3 until no object has changed clusters after a full cycle test of the whole data set.

Like the k-means algorithm the k-modes algorithm also produces locally optimal solutions that are dependent on the initial modes and the order of objects in the data set. In our current implementation of the k-modes algorithm we include two initial mode selection methods. The first method selects the first k distinct records from the data set as the initial k modes. The second method is implemented with the following steps.

1. Calculate the frequencies of all categories for all attributes and store them in a category array in descending order of frequency as shown in figure 1. Here, ci,j denotes category i of attribute j and f (ci,j) ≥ f (ci+1,j) where f (ci,j) is the frequency of category ci,j .

2. Assign the most frequent categories equally to the initial k modes. For example in figure 1, assume k = 3. We assign Q1 = [q1,1 = c1,1, q1,2 = c2,2, q1,3 = c3,3, q1,4 = c1,4],

2.2.7.3 k-prototype Clustering

The algorithm is based on the k-means paradigm but removes the

numeric data only limitation whilst preserving its efficiency. The algorithm clusters

objects with numeric and categorical attributes in a way similar to k-means. Because

objects are clustered against k prototypes instead of k means of clusters, called as k-prototyoe algprithm.

K-prototype is used to dynamically update the k

prototypes in order to maximise the intra cluster similarity of objects. The object

similarity measure is derived from both numeric and categorical attributes. When

applied to numeric data the algorithm is identical to k-means.

**Mathematical Preliminaries**

Let X = {X 1 ,X 2 ,...,X n } denote a set of n objects and X i = [x i1 ,x i2 ,...,x im ] be an object

represented by m attribute values. Let k be a positive integer. The objective of

clustering X is to find a partition which divides objects in X into k disjoint clusters.

For a given n, the number of possible partitions is definite but extremely large 1 .

It is impractical to investigate every partition in order to find a better one for a

classification problem. A common solution is to choose a clustering criterion to

guide the search for a partition. A clustering criterion is called a cost function below.

2.1

Cost Function

The widely used cost function is the trace of the within cluster dispersion matrix .

One way to define this cost function is

Eq 2.1

Here, Q l = [ q l 1 , q l 2 , ... , q lm ] is the representative vector or prototype for cluster

l, and y il is an element of a partition matrix Y nxl 9 . d is a similarity measure often

defined as the square Euclidean distance.

Y has the following two properties, (1) 0 ≤ y il ≤ 1 and (2) ∑ y il = 1 . Y is called a

l = 1

hard partition if y il ∈{0,1} . Otherwise, it is a fuzzy partition 2 . In a hard partition,

y il = 1 indicates that object X i is assigned to cluster l by Y.

The inner term E l = ∑ y il d ( X i , Q l ) in Eq. (2.1) is the total cost of assigning X to

i = 1

cluster l, i.e., the total dispersion of objects in cluster l from its prototype Q l . E l is

minimised if

Eq 2.2

where n l = ∑ y il is the number of objects in cluster l.

i =1

When X has categorical attributes, we can introduce a similarity measure as

Eq 2.3

where δ (p,q)=0 for p=q and δ (p,q)=1 for p≠q. x ijr and q ljr are values of numeric

attributes, whereas x ijc and q ljc are values of categorical attributes for object i and the

prototype of cluster l. m r and m c are the numbers of numeric and categorical

attributes. γ l is a weight for categorical attributes for cluster l.

We can rewrite E l as

eq 2.4

where E lr is the total cost on all numeric attributes of objects in cluster l. E lr is

minimised if q ljr is calculated by Eq. (2.2).

Let C j be the set containing all unique values in the categorical attribute j and

5p ( c j ∈ C j | l ) the probability of value c j occurring in cluster l. E lc in Eq. (2.4) can

be rewritten as

eq 2.5

where n l is the number of objects in cluster l

### **Similiarity Measure**

### The cost function Eq. (2.6) is defined on Eq. (2.3), which is a combined similarity

### measure on both numeric and categorical attributes between objects and cluster

### prototypes. The similarity measure on numeric attributes is the square Euclidean

### distance whereas the similarity measure on categorical attributes is the number of

### mismatches between objects and cluster prototypes. Weight γ l is introduced to avoid

### favouring either type of attribute.

### The influence of weight γ l in clustering can be illustrated by Figure 1. Assume

### the triangles and diamonds represent a set of objects described by a categorical and

### two numeric attributes. Triangle and diamond represent two values of the categorical

### attribute whereas numeric attribute values are reflected by locations of the objects.

### These objects are partitioned into two clusters.

000000000000

### If γ l =0, clustering only depends on numeric attributes, i.e., locations of the

### objects. The result will be the two clusters separated by the vertical dashed line.

### If γ l > 0, then object c may change to the right cluster because it is close to that

### cluster and its categorical value is the same as that of the majority of objects in that

### cluster. Similarly, object d may change to the left cluster. However, object a may still

### stay in the left cluster because it is too far to the right, even though it has a

### categorical value as same as that of the majority objects in that cluster. Similarly,

### object e may still be in the right cluster. Object b becomes uncertain, depending on

### whether γ l is biased towards numeric or categorical attributes. If γ l is biased to the

### categorical attribute, object b may change to the right cluster. Otherwise, it can stay

### in the left one.

### The choice of γ l is dependent on distributions of numeric attributes. Generally

### speaking, γ l is related to σ l , the average standard deviation of numeric attributes in

### cluster l. In practice, σ l can be used as a guidance to determine γ l . However,

### since σ l is unknown before clustering, the overall average standard deviation σ of

### numeric attributes can be used for all σ l . In an iterative algorithm, σ l can be

### calculated from a preceding clustering result. In Section 4.2 we will present some

### simulations on γ l .

### **K-prototype Algorithm**

### 2.1.1 Hamiltonian Path

A Hamiltonian path is a path in which each vertex is visited exactly once is known as Hamiltonian path. A graph which contains a Hamiltonian path is called a traceable graph. A graph is said to be Hamiltonian-connected if there exists a Hamiltonian path for every pair of vertices.

A Hamiltonian cycle, Hamiltonian circuit, vertex tour or graph cycle is a cycle in which each vertex is visited exactly once (except for the vertex that is both the source and destination, which is visited twice). A graph containing a Hamiltonian cycle is known as Hamiltonian graph.

Directed graphs also have Hamiltonian path in which each edge of a path or cycle can only be traced in a single direction.

A Hamiltonian path can be converted into Hamiltonian circuit by using edge decomposition which converts it into a Hamiltonian circuit. A Hamiltonian cycle can be converted to a Hamiltonian path by removing one of its edges, but a Hamiltonian path can be converted to Hamiltonian cycle by connecting its adjacent end points. All Hamiltonian graphs are biconnected, but it is not necessary that biconnected graph is Hamiltonian [25].

An Eulerian graph G is a graph in which every vertex has even degree. An Eulerian graph necessarily has an Euler tour in which there is a closed walk which passes through each edge of G exactly once. So, Euler tour corresponds to a Hamiltonian cycle in the line graph L(G) and it can be said that line graph is Hamiltonian graph. Line graphs may have other Hamiltonian cycles that do not correspond to Euler tours, and in particular the line graph L(G) of every Hamiltonian graph G is itself Hamiltonian, regardless of whether the graph G is Eulerian [24]. A tournament (with more than two vertices) is Hamiltonian if and only if it is strongly connected. The number of different Hamiltonian cycles in a complete undirected graph on n vertices is (n − 1)! / 2 and in a complete directed graph on n vertices is (n − 1)! [25].

## 2.2 Genetic Algorithm

Genetic algorithms (GAs) deals with the concept of the survival of the fittest among the species which are generated by randomly changing the genes in the chromosomes in the evolutionary biological species. To solve some real life problem using GA, two of the main requirements are to be satisfied

(1) A string known as a chromosome in terms of GA which can represent a solution of the solution space, and

(2) A fitness function is used to measure the goodness of a solution

A simple GA works by randomly generating population of chromosome, which can be referred as gene pool and then applying different operators to create new and better populations as their successive generations. The first operator is reproduction where chromosome is copied to the next generation with the probability based on their fitness function value [12]. The second operator is crossover where random pairs are selected and mated to create new chromosome. The third operator, mutation, which is a random alteration of the gene position in the chromosome. The reproduction together with crossover is the most powerful process in the GA .

Mutation diversifies the search space and protects from loss of genetic material that can be caused by reproduction and crossover [9]. So, the probability of applying mutation is set to be very low, whereas the probability of crossover is set to be very high.

### 2.2.1 Genetic Coding

To apply GA for any optimization problem, the most important step is encoding solutions as feasible chromosomes so that the crossovers of feasible chromosomes result in more feasible chromosomes. The techniques for encoding solutions require special care, a good intelligence and depend on the problem statement. For the TSP, solution is typically represented by chromosome of length equal to the number of nodes in the problem. Each gene of a chromosome represents a node such that no node can appear twice in the same chromosome. There are mainly two mostly used representation methods for representing tour of the TSP – path representation and adjacency representation. The path representation for a tour simply lists the label of nodes. For example, let {1, 2, 3, 4, 5} be the labels of nodes in a 5 node instance, then a tour {1→ 3→4→ 2→ 5 →1} may be represented as (1, 3, 4, 2, 5) [11].

### 2.2.2 Fitness Function

The GAs are used for maximization problem. For minimization problem, one way of defining a fitness functions F(x) = 1/f(x) where f(x) is a objective function. As TSP is a minimization problem, we consider this fitness function F(x) = 1/f(x) where f(x) calculates cost of the tour represented by a chromosome.

### 2.2.3 Reproduction Operator

Reproduction/Selection process is a way in which chromosomes are copied into next generation mating pool according to the probability associated with their fitness value. When we assign the higher portion of the highly fit chromosomes to the next generation, reproduction mimics the Darwinian survival-of-the-fittest in the natural world. In natural population, fitness is determined by a creature‘s ability to survive predators, pestilence, and other obstacles to adulthood and subsequent reproduction. No new chromosome is produced in this phase. The reproduction operator which is commonly used is the proportionate reproduction operator, where a string is selected for the mating pool with a probability proportional to its fitness value.

### 2.2.4 Sequential Constructive Crossover Operator (SCX)

The search space for the solution is generated by creating new chromosomes from existing ones. Crossover is the most important search process. Different crossover techniques are used to generate new children like one-point crossover, two-point crossover. Initially in a crossover a pair of parents is selected from the pool of chromosome randomly. In second step, a point usually called crossover site is selected randomly, and the information of the two parent chromosome after the crossover site are interchanged resulting into two new offspring.

The algorithm for the SCX is as follows:

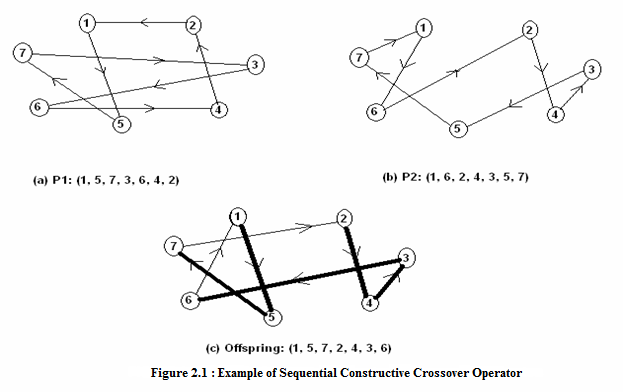
**Step 1**: - Start from node 1 (i.e., current node p =1).

**Step 2**: - Sequentially search both of the parent chromosomes and consider the first legitimate node (the node that is not yet visited) appeared after node p in each parent. If no legitimate node after node p is present in any of the parent, search sequentially the nodes {2, 3, …, n} and consider the first legitimate node, and go to Step 3.

**Step 3**: Suppose the node α and the node β are found in 1st and 2nd parent respectively, then for selecting the next node go to Step 4.

**Step 4:** If cpα <cpβ, then select node α, otherwise, node β as the next node and concatenate it to the partially constructed offspring chromosome. If the offspring is a complete chromosome, then stop, otherwise, rename the present node as node p and go to Step 2.

Let us illustrate the SCX through the example given as cost matrix in Table 2.1. Let a pair of selected chromosomes be P1: (1, 5, 7, 3, 6, 4, 2) and P2: (1, 6, 2, 4, 3, 5, 7) with values 312 and 331 respectively [12].



**Table 2.1: The Cost Matrix**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Node | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 1 | 999 | 75 | 99 | 9 | 35 | 63 | 8 |
| 2 | 51 | 999 | 86 | 46 | 88 | 29 | 20 |
| 3 | 100 | 5 | 999 | 16 | 28 | 35 | 28 |
| 4 | 20 | 45 | 11 | 999 | 59 | 53 | 49 |
| 5 | 86 | 63 | 33 | 65 | 999 | 76 | 72 |
| 6 | 36 | 53 | 89 | 31 | 21 | 999 | 52 |
| 7 | 58 | 31 | 43 | 67 | 52 | 60 | 999 |

### 2.2.5 Offspring by Two Other Crossover Operators

The other crossover operators used for generating offspring are – edge recombination crossover (ERX) and generalized N-point crossover (GNX) for producing offspring which uses same pair of parents P1 and P2 as used in SCX.

For ERX, the edge table of the example discussed above is shown in Table 2.2 [12]. The new offspring is initialized with node 1, 5, 2, 6 and 7 are the candidates for the next node. Node 6 has three edges and therefore not considered. Assume node 5 is randomly chosen. Node 5 is connected to nodes 7 and 3 so node 7 is chosen next. Node 7 only has an edge to node 3 so node 3 is chosen next. Node 3 has edges to nodes 4 and 6 both of which have two edges. Suppose the node 4 is randomly chosen then the node 4 has edges to the nodes 6 and 2 both of which have one edge left. Next randomly choose node 6 which has an edge to node 2 of course this is the last node to be selected, so node 2 is chosen next. The resulting offspring may be (1, 5, 7, 3, 4, 6, 2).

**Table 2.2: Edge Table for the Parents P1 and P2**

|  |  |  |  |
| --- | --- | --- | --- |
| **Node** | **Edge list** | **Node** | **Edge list** |
| **1** | 5, 2, 6, 7 | **5** | 1, 7, 3 |
| **2** | 4,1,6 | **6** | 3,4,1,2 |
| **3** | 7, 6, 4, 5 | **7** | 5, 3, 1 |

For GNX, let’s take N=2 and consider same parents P1: (1, 5, 7, 3, 6, 4, 2) and P2: (1, 6, 2, 4, 3, 5, 7) and G2X with cross points 3 and 5. Suppose the order in which the segments are tested is (2, 3, 1). Then the 2nd Segment of P will be inserted whole, giving the proto-child (x, x, x, 3, 6, x, x). Nodes in the 3rd segment from P2 will then be tested in a random order. Both the city 5 and 7 will be accepted, giving segment of P will be inserted whole, giving the proto-child (x, x, x, 3, 6, x, x). Nodes in the 3rd segment from p2 will be tested in a random order. So the final child may be given by (1, 4, 2, 3, 6, 5, 7) [12].

### 2.2.6 Survivor Selection

Survivor selection method is used for selecting population for next generating after the crossover operation. This survivor selection method of GA considers two kinds of chromosomes for the next generation: (1) parents in current population of size n, and (2) offspring that are generated by crossover of size n. The survivor selection method combines chromosomes in (1) and (2), sorts them in ascending order according to their fitness, and selects the first m chromosomes for the next generation. In worst case, all the parents which were used in present generation will survive and move to the next generation.

### 2.2.7 Mutation Operator

The mutation operator randomly selects a gene and changes that gene with another gene, thus modifying information. Mutation is basically used because without mutation some aspect of genetic material could be lost forever as less fit chromosome are discarded in next generation from the fact that as the less fit members of successive generations are discarded; some aspects of genetic material could be lost forever. By performing occasional random changes in the chromosomes, mutation ensure that new parts of the search space are reached, which reproduction and crossover alone couldn‘t fully guarantee. In doing so mutation ensures that no important features are completely lost, thus maintaining the diversity. For the TSP, the classical mutation operator does not work [10]. For this investigation, we have considered the reciprocal exchange mutation that selects two nodes randomly and swaps them.

### 2.2.8 Control Parameters

These are the parameters that govern the GA search process. Some of them are:

**Population size**: - It determines the total number of chromosome in the search space.

**Crossover probability**: - It specifies the probability of crossover occurring between two chromosomes.

**Mutation probability**: - It specifies the probability of doing mutation.

**Termination criteria**: - It specifies when to terminate the genetic search.

## 2.3 Particle Swarm Optimization (PSO) for Network Optimization

One of the inspiration from which PSO was developed was bird flocks which considers birds as a particle moving in a flock chasing for food. Each bird has some velocity and they continuously change their positions to get nearest to the food. Basically, particle swarms is used as a means to simulate social behavior of bird flocks by using computer-based model.

Swarm intelligence is an important method based on the collective behavior of decentralized and self-organized systems in computational intelligence [6]. It consists of a population which simulates the behavior of birds in the real world. There are many swarm intelligence optimization algorithms, such as particle swarm optimization, genetic algorithms, bee colony algorithm, ant colony optimization, differential evolution, fish -warm algorithm, etc. Due to the easy implementation and simple concept, PSO has gained much attention and been successfully applied in a variety of fields especially for optimization problems.

Goldberg developed a PSO algorithm for the TSP where the idea of distinct velocity operators is introduced. The velocity operators are defined according to the possible movements a particle is allowed to do. This algorithmic proposal obtained very promising results [4].

### 2.3.1 Classical PSO

The basic principles of PSO are very simple. A set of moving particles is initially thrown inside the search space. Each of these particles has the following features:

1. It has a position and a velocity
2. It knows its position
3. It knows its neighbors and best previous position( personal best)

At each step, the particle can move in three different directions as described below:

1. To follow its own way
2. To go towards its best previous position
3. To go towards the best neighbor’s best previous position, or towards the best neighbor

The decision depends upon three parameters:

1. how much the particle trusts itself
2. how much it trusts its experience
3. how much it trusts its neighbors

These parameters are usually randomly chosen, at each time step, in given intervals. Also, the neighborhood can be defined in many ways like:

1. Physical neighborhood, which takes distances into account. In practice, at each time step distances are recomputed, which is quite costly, but some techniques need this information.
2. Social neighborhood, which just takes relationships into account. In practice, for each particle, neighborhood is defined as a list of particles and does not change. Note that, when the process converges, a social neighborhood becomes a physical one.

### 2.3.2 Particle Swarm Optimization

The PSO is a optimization algorithm in which a given problem is solved by individual particles working together. The best thing about PSO algorithm is its simplicity. The PSO can be implemented in very small code using some of the mathematical operators, making it best for limited-memory and environment where there is computation constraint.

A basic variant of the PSO algorithm works by having a population (called a swarm) of candidate solutions (known as 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. When improved positions are being discovered these will then come to guide the movements of the swarm [20]. The process is repeated and by doing so it is hoped, but not guaranteed, that a satisfactory solution will eventually be discovered. Formally, letbe thecost function which must be minimized. The functiontakes 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 for which for all in the search-space, which would mean is the global minimum [20]. Maximization can be performed by considering the function instead. PSO was presented under the inspiration of bird flock immigration during the course of finding food and then be used in the optimization problems. In PSO, each optimization problem solution is taken as a bird in the searching space and it is called “particle”. Every particle has a fitness value which is determined by target functions and it has also a velocity which determines its destination and distance. All particles search in the solution space for their best positions and the positions of the best particles in the swarm. PSO is initially a group of random particles (random solutions), and then the optimum solutions are found by repeated searching. In every iteration, a particle will follow two bests to renew itself: the best position found for a particle called pBest, the best position found for the whole swarm called gBest.

### 2.3.3 Basic PSO Algorithm

The basic flow for the PSO algorithm:

1. We begin by initializing random particle(s) (also called it population) within N-dimensional space.
2. As we have randomly initialized the population, so each particle will have a random velocity and thus location for each population.
3. Then fitness of each particle is evaluated according to the given problem. If fitness is found to be better than the particle best fitness (personal best), we will save this location for that particle as pBest (personal best). If the particle's fitness is better than the global best fitness (global best i.e. best among all the particles), we save location of this particle as gBest (global best).
4. Finally, we update the particle’s position and velocity and look at the next particle in the population.
5. If the global best fitness value meets the exit criteria (satisfactory criteria- as per the problem), we end the loop and provide the location as the solution to the given problem.

# CHAPTER 3

# PROPOSED METHOD

## 3.1 Travelling Salesman Problem Using PSO

This is basically to solve Travelling Salesman Problem in which we need to find the shortest route among all cities without visiting any city twice. Let take an example of 8 cities named 0, 1, 2, 3, 4, 5, 6 and 7. So, there are 88 (or, 16,777,216) possible combinations, but this PSO algorithm can find the solution computing less than 83 solutions.

The locations of the cities we have taken in our example are (30, 5), (40, 10), (29, 25), (40, 20), (19, 25), (20, 5), (9, 19), (9, 9).This is more like a circle: the solution can be easily checked by seeing a graph and compared the algorithm's solutions with. By performing some calculations, we have found the solution to be about 86.6299, which is the target value of our algorithm.

To simplify this example, starting or ending point doesn’t matter, moreover the direction of the tour travels also doesn’t matter. For example, a solution that looks like 23456701 is valid because it travels forward from 2 and around to 1. The solution 65432107 is equally valid because it just goes backward from 6 to 7.

The most important part of the PSO algorithm is the calculation of the velocity vector as it is this velocity vector which updates the particle position. So, we first added the global worst to the global variables. The velocity score is then calculated using this global worst.

In this example, only three variables can be experimented with:

1. COUNT\_OF\_PARTICLE - number of particles in the test.
2. MAX\_VELOCITY - maximum change allowed for velocity.
3. EPOCHS\_MAX – maximum number of iterations.

**Figure 3.1: Coordinates of Cities Location**

### 3.1.1 How it works?

Every particle has some position (x, y) and a velocity (vx, vy) with which it is moving through. Every particle has some tendency to move in direction same as before.

Every particle possesses some acceleration due to change in their velocity which depends upon two things:

1. Each particle will try to move towards the best location that it has found personally (also known as personal best-pBest).
2. Each particle will try to move toward the best location that any particle has found (also known as global best-gBest).

The strength with which the particles are attracted in each of these directions depends upon the parameters like pBest and gBest. PSO is used to either maximize or minimize the fitness function by updating the position in every loop. All paths from source to destination are considered as particles.

### 3.1.2 Procedure

For solving Travelling Salesman Problem using Particle Swarm Optimization ,we first choose some random paths(particles). In the example we are going to discuss we have taken 4 randomly generated paths(particle).We choose these paths by randomizing the initial path value( (0,1,2,3,4,5,6,7) for all the four particles).The only thing need to keep in mind while choosing these paths is that the path should be such that it should connect all points(cities) and not repeating any point(city) twice.

So

Initialize:

1. Path 10 1 2 3 4 5 6 7
2. Path 20 1 2 3 4 5 6 7
3. Path 30 1 2 3 4 5 6 7
4. Path 40 1 2 3 4 5 6 7

Now randomly rearrange each path(particle) 10 times. For rearranging these particles we select two cities for any path keeping in mind the two cities are different and swap them. We do this 10 times to increase the randomness. After randomly rearranging we get the following paths.

1. Path 12 4 3 6 5 7 0 1
2. Path 21 3 5 2 6 0 7 4
3. Path 31 3 5 2 6 0 7 4
4. Path 46 2 5 3 7 4 1 0

We calculate the distances of all these four paths(particles).

The distance comes out to be:

1. Path 1116.18432
2. Path 2180.60110
3. Path 3182.07424
4. Path 4185.11207

Now we will iterate until the maximum number of iterations has been reached or the target has been found. The maximum iteration count we have taken in this example is 10000. The target value taken by us is 86.63. If the result distance of any path comes out to be less than or equal to 86.63 we stop the iteration.

After every iteration the path value of every path changes due to change in velocity introduced. So we keep in account the personal best of each path (particle) .The personal best of each particle will be the best value (the minimum distance ) the particle has encountered after n iteration.

We also need to keep track of the global best of all the particle. Global best means the best value (minimum distance) achieved so far.

The next step we are going to do is to bubble sort the paths (particles) according to their personal best scores i.e from best to worst.

We now update the particle by changing the path (particle) value.We do this with the help a very important factor, velocity. We calculate the velocity by using the formulae

Value=(MAX\_VELOCITY\*(personal best of path( particle) )/worst result;

Here MAX\_VELOCITY is the maximum velocity allowed. We have taken MAX\_VELOCITY as 4.

Here worst result is the worst pBest among all paths.

This value is the indication of the no of changes to be allowed to all of the four paths so as to update them.

The values comes out to be:

1. Path 12
2. Path 22
3. Path 33
4. Path 43

Now we will again rearrange the four path according to the no of changes allowed.

After rearranging the paths comes out to be

1. Path 12 4 3 6 5 7 0 1 116.18436
2. Path 21 5 7 4 0 6 3 2 162.00802
3. Path 35 4 1 3 2 0 7 6 117.88545
4. Path 43 6 2 4 7 5 0 1 172.94305

Now these are the results after first iteration. We check whether the target value is achieved or not .We also take in account the maximum iterations.

The whole process is repeated till we reach our destination of finding the shortest path connecting all cities.

So we again bubble sort the paths (particles) and update the velocity. The updated velocity comes out to be:

1. Path 12
2. Path 22
3. Path 32
4. Path 42

Updating the paths (particles), the paths (particles) comes out to be

1. Path 12 3 4 7 6 5 0 1 110.23192
2. Path 22 4 3 6 5 7 0 1 116.18436
3. Path 32 4 1 3 5 7 0 6 114.15432
4. Path 46 7 4 5 3 9 1 2 138.37124

Again for the example we have taken the target value is not reached after the 2nd iteration. So again updating the path by applying changes to path with the velocity. After iterating 800 times we get the result in the 801 iteration.

The updated velocity comes out to be:

1. Path 12
2. Path 23
3. Path 32
4. Path 42

The updated path will be:

1. Path 12 3 4 7 6 5 0 1 110.23192
2. Path 23 4 5 7 6 2 0 1 131.82403
3. Path 37 0 1 2 3 4 5 6  86.62995
4. Path 42 4 3 6 5 7 0 1 116.18436

Here we get the personal best of the path (particle) less than the target, we will terminate and finally get the shortest route connecting all points (cities).

The shortest path for out example in

Route 7 0 1 2 3 4 5 6

The distance for the shortest path is 86.6299.

We analyze the algorithm by changing the count of the no of paths and the velocity\_max i.e. the maximum allowable change.

We get the following result.

**Case 1: COUNT\_OF\_PARTICLE = 10, MAX\_VELOCITY = 4, EPOCHS\_MAX = 10000.**

Route 2 3 4 6 5 7 0 1  99.9458

Route 2 7 3 4 6 5 0 1  132.2345

Route 6 4 7 3 5 0 1 2  161.0345

Route 4 7 0 5 2 6 1 3  178.4356

Route 3 6 1 5 0 4 7 2  194.8090

Route 4 6 0 7 2 1 5 3  148.6579

Route 7 0 1 2 3 4 5 6  86.4536

Route 5 2 3 6 7 0 1 4  139.6543

Route7 2 3 6 0 4 5 1  171.7685

Route 2 3 0 4 6 1 5 7  179.8768

Changes for particle 1 2

Changes for particle 2 1

Changes for particle 3 4

Changes for particle 4 3

Changes for particle 5 3

Changes for particle 6 2

Changes for particle 7 4

Changes for particle 8 1

Changes for particle 9 4

epoch number: 87

Target reached.

Shortest Route: 7, 0, 1, 2, 3, 4, 5, 6, Distance: 86.4536

**Case 2:COUNT\_OF\_PARTICLE = 10, MAX\_VELOCITY = 8, EPOCHS\_MAX = 10000.**

Route: 0, 7, 5, 6, 4, 3, 2, 1, Distance: 99.6786

Route: 1, 0, 7, 6, 5, 4, 3, 2, Distance: 86.6345

Route: 6, 4, 1, 5, 2, 3, 0, 7, Distance: 161.5426

Route: 5, 2, 3, 0, 4, 1, 6, 7, Distance: 172.4235

Route: 1, 6, 4, 0, 2, 3, 5, 7, Distance: 162.5345

Route: 0, 1, 3, 2, 7, 5, 6, 4, Distance: 136.65456

Route: 0, 3, 6, 5, 1, 2, 7, 4, Distance: 165.6578

Route: 6, 5, 4, 7, 0, 3, 1, 2, Distance: 133.6764

Route: 7, 3, 1, 2, 4, 6, 5, 0, Distance: 136.4567

Route: 6, 5, 7, 3, 1, 4, 2, 0, Distance: 155.4568

Changes for particle 1: 5

Changes for particle 2: 3

Changes for particle 3: 6

Changes for particle 4: 8

Changes for particle 5: 2

Changes for particle 6: 5

Changes for particle 7: 7

Changes for particle 8: 6

Changes for particle 9: 8

epoch number: 223

Target reached.

Shortest Route: 1, 0, 7, 6, 5, 4, 3, 2, Distance: 86.6325

**Increasing the velocity has very less effect.**

### 3.1.3 Program Module for PSO applied on TSP

**Module for applying basic PSO Algorithm:**

private static void mainalgo()

{

Particle aParticle = null;

int epoch = 0;

boolean done = false;

initialization();

while(!done)

{

// two conditions which can end this loop are:

// the number of maximum epochs or iterations allowed has been reached, or,

// the Target is reached.

if(epoch < MAX\_EPOCHS){

for(int i = 0; i < PARTICLE\_COUNT; i++)

{

aParticle = particles.get(i);

System.out.print("Route: ");

for(int j = 0; j < CountOf\_City; j++)

{

System.out.print(aParticle.data\_Item(j) + ", ");

}

TotalDistance(i);

if(aParticle.pBest() <= TARGET){

done = true;

}

}

Sort(); // sorting of particles according to their pBest scores, best to worst.

newvel\_ocity();

update\_particles();

System.out.println("epoch number: " + epoch);

epoch++;

}

else{

done = true;

}

}

return;

}

**Module for initializing particles with random values(choosing random path):**

private static void initialization()

{

for(int i = 0; i < PARTICLE\_COUNT; i++)

{

Particle newParticle = new Particle();

for(int j = 0; j < CountOf\_City; j++)

{

newParticle.data\_Item(j, j);

}

particles.add(newParticle);

for(int j = 0; j < 10; j++)

{

randomly\_Arrange(particles.indexOf(newParticle));

}

TotalDistance(particles.indexOf(newParticle));

}

return;

}

**Module to get velocity:**

private static void newvel\_ocity()

{

double worstResults = 0;

double vValue = 0.0;

// after sorting, worst element will be the last element in list.

worstResults = particles.get(PARTICLE\_COUNT - 1).pBest();

for(int i = 0; i < PARTICLE\_COUNT; i++)

{

vValue = (V\_MAX \* particles.get(i).pBest()) / worstResults;

if(vValue > V\_MAX){

particles.get(i).vel\_ocity(V\_MAX);

}else if(vValue < 0.0){

particles.get(i).vel\_ocity(0.0);

}else{

particles.get(i).vel\_ocity(vValue);

}

}

return;

}

**Program module to update particle (generating new path ):**

private static void update\_particles()

{

// Best value of the particle is at index 0, so start from the second best.

for(int i = 1; i < PARTICLE\_COUNT; i++)

{

// The particle wil need more changes, when the vel\_ocity of the particle is higher.

int changes = (int)Math.floor(Math.abs(particles.get(i).vel\_ocity()));

System.out.println("Changes for particle " + i + ": " + changes);

for(int j = 0; j < changes; j++){

if(new Random().nextBoolean()){

randomly\_Arrange(i);

}

}

// Update the pBest value.

getTotalDistance(i);

}

return;

}

**Program module to get the best solution:**

private static void printoptimalSolution()

{

if(particles.get(0).pBest() <= TARGET){

// Print it.

System.out.println("Target reached.");

}else{

System.out.println("Target not reached");

}

System.out.print("Shortest Route: ");

for(int j = 0; j < CountOf\_City; j++)

{

System.out.print(particles.get(0).data\_Item(j) + ", ");

}

return;

}

**Program module to get total distance of a path (particle):**

private static void TotalDistance(final int index)

{

Particle thisParticle = null;

thisParticle = particles.get(index);

thisParticle.pBest(0.0);

for(int i = 0; i < CountOf\_City; i++)

{

if(i == CountOf\_City - 1){

thisParticle.pBest(thisParticle.pBest() + getDistance(thisParticle.data\_Item(CountOf\_City - 1), thisParticle.data\_Item(0))); // Full trip.

}else{

thisParticle.pBest(thisParticle.pBest() + getDistance(thisParticle.data\_Item(i), thisParticle.data\_Item(i + 1)));

}

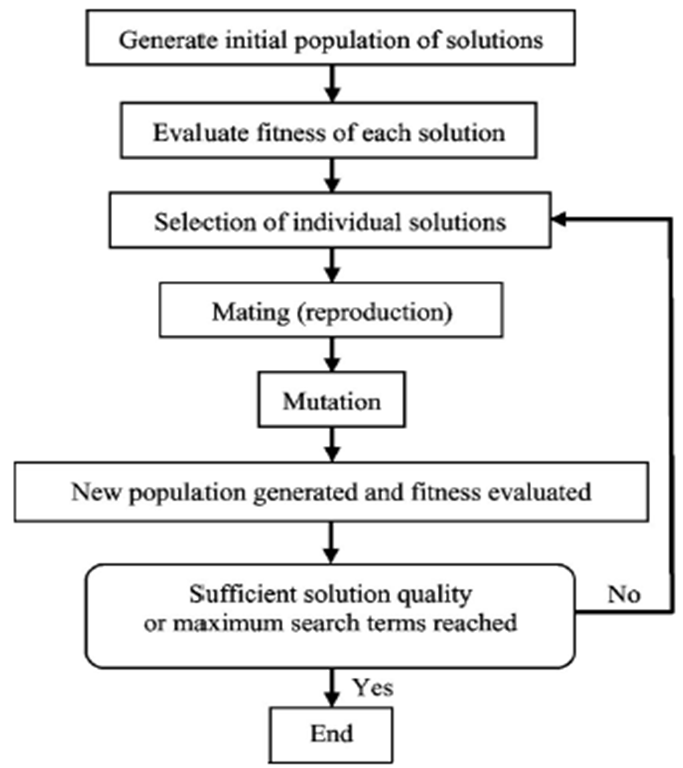
}

return;

}

## 3.2 Genetic Algorithm for Solving TSP

Genetic algorithm is a search heuristic used in the field of artificial intelligence that explains the process of natural evolution. Genetic Algorithm belongs to a class of evolutionary algorithm.GA is applied on a set of chromosomes which are encoded as a population of chromosomes, then a fitness function is used for the evaluation of the fitness of each chromosome, after that offspring is generated through the process of selecting a set of chromosomes known as selection, then selected chromosomes undergoes crossover to create an offspring and finally mutation. After the termination of GA, an optimal solution is found. If the optimal solution is not found depending upon the threshold value, the algorithm is run again with new population. Flowchart for proposed GA is described below-



**Figure 3.2: Flowchart of Genetic Algorithm**

### 3.2.1 Implementation of Proposed Algorithm

We have taken 15 cities and the coordinates of those cities are shown in Fig 3.3:

**Figure 3.3: Coordinates of the 15 Cities for Solving TSP Using GA**

The distance-matrix created for the distance between different cities is symmetric i.e. if we moves from city 1 to city 4 than the distance will be same if we move from city 4 to city 1. Due to this half of the entries from the matrix are omitted.

**Table 3.1: Distance Matrix of 15 Cities**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| CITY | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| 1 | 0 | 29 | 82 | 46 | 68 | 52 | 72 | 42 | 51 | 55 | 29 | 74 | 23 | 72 | 46 |
| 2 |  | 0 | 55 | 46 | 42 | 43 | 43 | 23 | 23 | 31 | 41 | 51 | 11 | 52 | 21 |
| 3 |  |  | 0 | 68 | 46 | 55 | 23 | 43 | 41 | 29 | 79 | 21 | 64 | 31 | 51 |
| 4 |  |  |  | 0 | 82 | 15 | 72 | 31 | 62 | 42 | 21 | 55 | 51 | 43 | 64 |
| 5 |  |  |  |  | 0 | 74 | 23 | 52 | 21 | 46 | 82 | 58 | 46 | 65 | 23 |
| 6 |  |  |  |  |  | 0 | 61 | 23 | 55 | 31 | 33 | 37 | 51 | 29 | 59 |
| 7 |  |  |  |  |  |  | 0 | 42 | 23 | 31 | 77 | 37 | 51 | 46 | 33 |
| 8 |  |  |  |  |  |  |  | 0 | 33 | 15 | 37 | 33 | 33 | 31 | 37 |
| 9 |  |  |  |  |  |  |  |  | 0 | 29 | 62 | 46 | 29 | 51 | 11 |
| 10 |  |  |  |  |  |  |  |  |  | 0 | 51 | 21 | 41 | 23 | 37 |
| 11 |  |  |  |  |  |  |  |  |  |  | 0 | 65 | 42 | 59 | 61 |
| 12 |  |  |  |  |  |  |  |  |  |  |  | 0 | 61 | 11 | 55 |
| 13 |  |  |  |  |  |  |  |  |  |  |  |  | 0 | 62 | 23 |
| 14 |  |  |  |  |  |  |  |  |  |  |  |  |  | 0 | 59 |
| 15 |  |  |  |  |  |  |  |  |  |  |  |  |  |  | 0 |

### 3.2.2 Initial Population

A unique random number generator function is used to create initial population of chromosomes. The table 3.2 shows the initial population of chromosomes. The initial population comprises of ten chromosomes. Each chromosome is made up of genes used to represent the number assigned to a city, where each chromosome denotes the sequence in which cities have to be traversed.

**Table 3.2: Chromosome Path value**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Chromosome1 | 1 | 4 | 13 | 3 | 8 | 2 | 5 | 15 | 7 | 10 | 14 | 12 | 6 | 9 | 11 |
| Chromosome2 | 3 | 14 | 13 | 2 | 9 | 10 | 5 | 7 | 1 | 15 | 6 | 12 | 8 | 11 | 4 |
| Chromosome3 | 1 | 15 | 3 | 7 | 14 | 11 | 9 | 2 | 13 | 5 | 12 | 4 | 8 | 10 | 6 |
| Chromosome4 | 4 | 12 | 14 | 13 | 5 | 9 | 11 | 8 | 1 | 3 | 10 | 2 | 6 | 7 | 15 |
| Chromosome5 | 11 | 2 | 9 | 5 | 13 | 14 | 3 | 12 | 8 | 1 | 15 | 6 | 4 | 10 | 7 |
| Chromosome6 | 3 | 10 | 7 | 13 | 11 | 2 | 9 | 4 | 15 | 12 | 6 | 5 | 14 | 1 | 8 |
| Chromosome7 | 11 | 5 | 2 | 9 | 15 | 13 | 7 | 8 | 4 | 1 | 3 | 12 | 6 | 10 | 14 |
| Chromosome8 | 3 | 4 | 13 | 14 | 11 | 7 | 10 | 2 | 8 | 15 | 1 | 5 | 9 | 12 | 6 |
| Chromosome9 | 10 | 11 | 7 | 8 | 15 | 1 | 5 | 9 | 12 | 4 | 14 | 6 | 2 | 13 | 3 |
| Chromosome10 | 10 | 11 | 4 | 7 | 12 | 1 | 6 | 3 | 9 | 5 | 15 | 14 | 13 | 8 | 2 |

### 3.2.3 Fitness Value

Fitness function is used to check for the extent of goodness of a chromosome it means that it gives an idea about how good the chromosome is? The length of a chromosome is used as basic criteria for a good chromosome. Chromosomes are created by performing some calculations. Each chromosome is created and then its fitness function is calculated.

### 3.2.4 Selection

Chromosome with smallest fitness value is selected in the Selection method. We have used the roulette wheel selection method.

### 3.2.5 Roulette Wheel Selection Method

Roulette wheel selection method is used for selecting chromosomes from the given population of chromosomes because best chromosomes should be selected to generate new offspring. Parents are selected according to their fitness. The chromosomes are placed on a wheel according to their fitness value. A marble is thrown to select the chromosome and a chromosome with best fitness value will be selected more number of times.

### 3.2.6 Crossover

Crossover is used to generate offspring from the existing parents. Different crossover techniques are used to generate new offspring like 1-point crossover in which an offspring is generated by interchanging gene at one location within two parents. Another method used for crossover is 2-point crossover in which randomly two positions in the chromosomes are chosen and then replace the gene with each other in both chromosomes.

### 3.2.7 Mutation

Mutation is further applied to form a new generation. Different techniques are used for performing mutation like swap adjacent mutation in which adjacent genes a chromosome are interchanged to generate a new chromosome. We applied interchange mutation in which it randomly selects two genes in chromosome and interchange them to generate a new

chromosome. Mutation is applied because there is more probability of finding better chromosome.

### 3.2.8 Termination and Result

The process is terminated as soon as the best tour is found. The tour after completing the number of iterations the best tour will be obtained and the process will be terminated. In figure 4.5 the tour with minimum distance is shown. The minimum distance comes out to be 291 for the problem of 15 cities.

# CHAPTER 4

# EXPERIMENTAL RESULT AND ANALYSIS

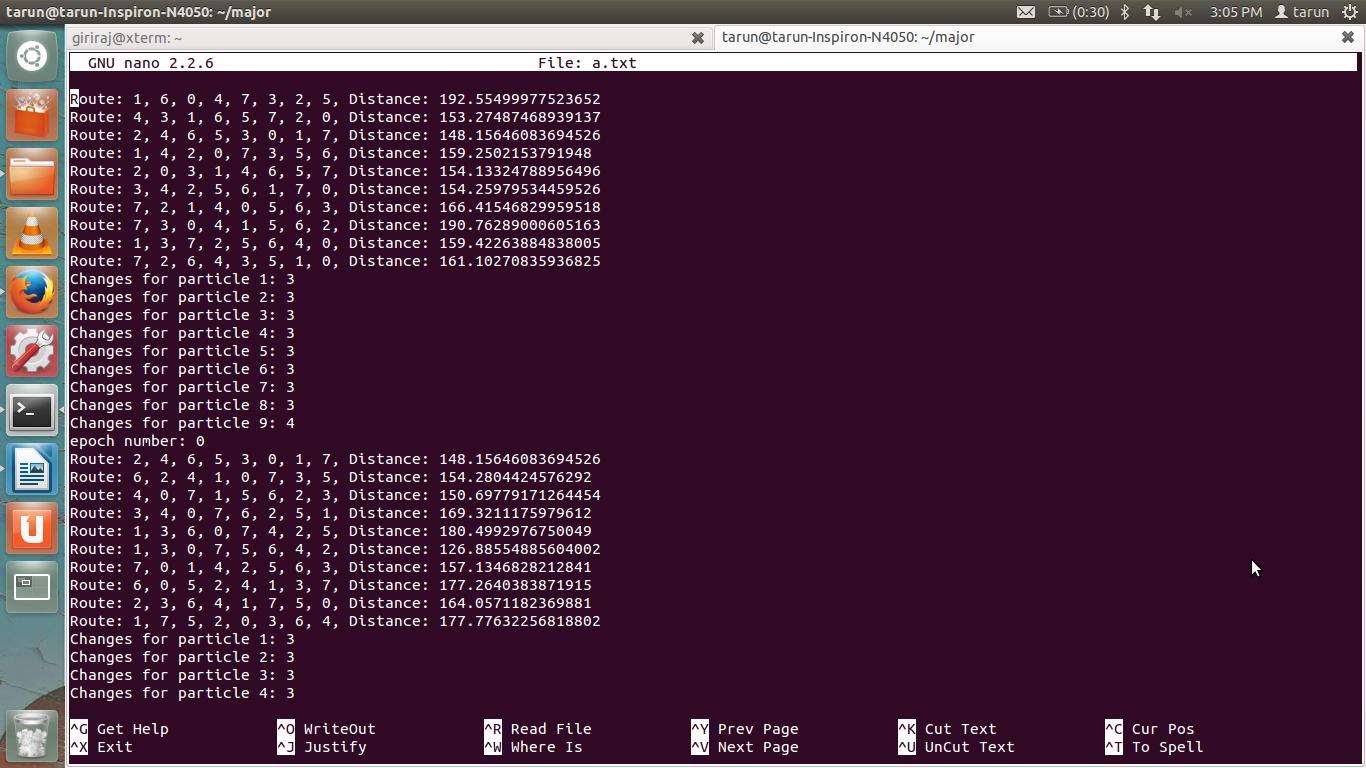
## 4.1 Result Obtained after Application of PSO on TSP

The coordinates of the cities used in our problem were 0-(30,5), 1-(40,10), 2-(40,20), 3-(29,25), 4-(19,25), 5-(9,19), 6-(9,9), 7-(20,5)

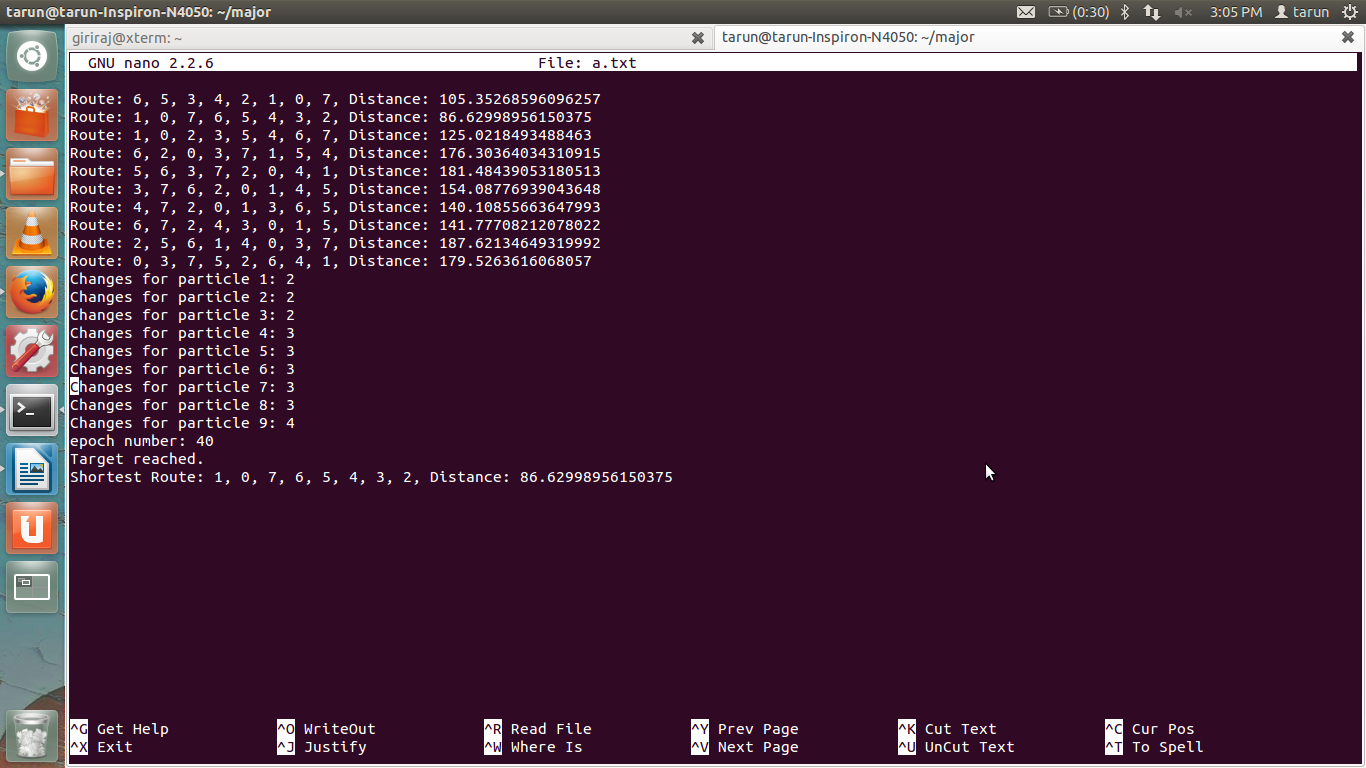
After applying PSO on TSP the shortest route comes out to be 7-0-1-2-3-4-5-6. The shortest distance is 86.6299.

**Figure 4.1: Path obtained after applying TSP using PSO**

**4.1.1 Output on Running PSO on Travelling Salesman Problem**



**Figure 4.2: Snapshot of the output of PSO for solving TSP**



**Figure 4.3: Snapshot of the output of PSO for solving TSP**

## 4.2 Result Obtained after Application of GA on TSP

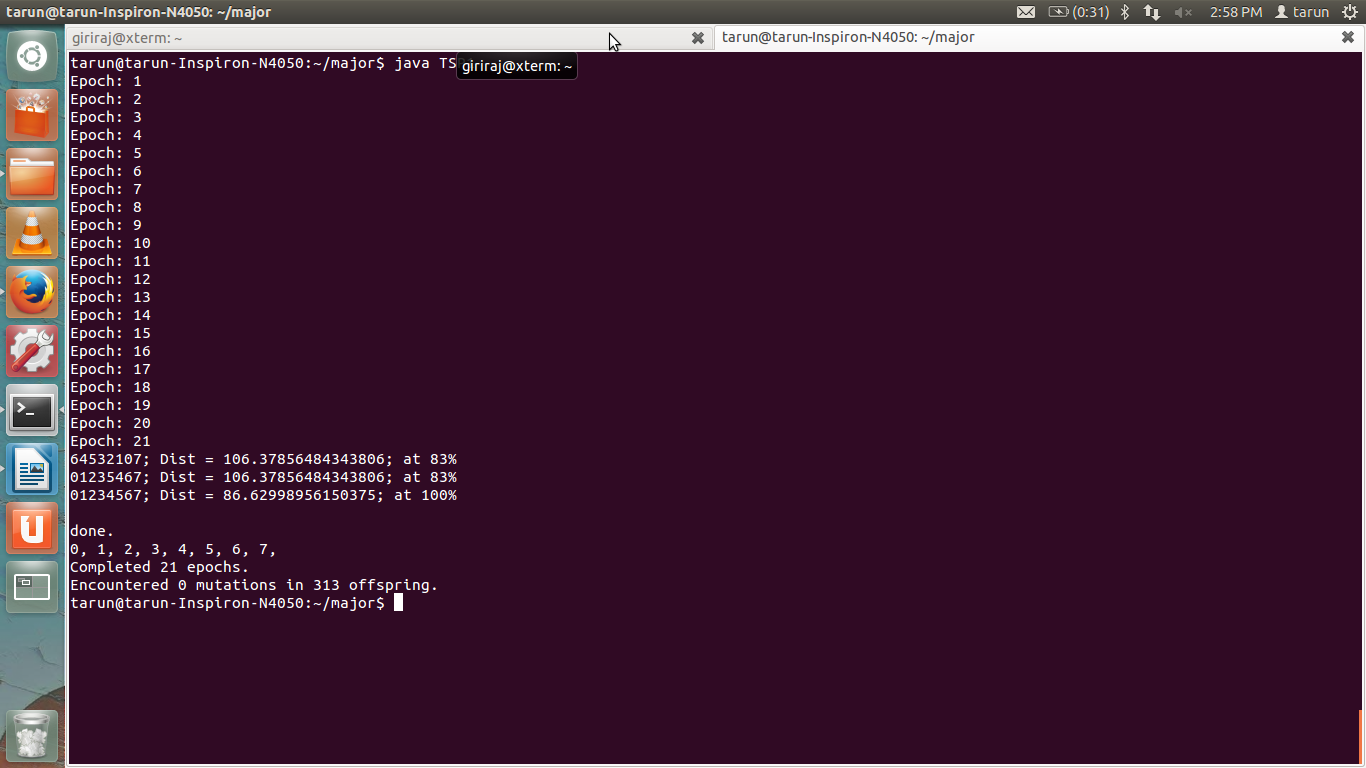
We have taken 15 cities and the coordinates of those cities are

**Figure 4.4: Coordinates of cities for solving TSP**

After applying PSO on TSP the shortest route comes out to be

**Figure 4.5: Shortest Path after applying GA on TSP**

### 4.2.1 Output on Running Genetic Algorithm on Travelling Salesman Problem



**Figure 4.6: Snapshot of the output of GA for solving TSP**

# CHAPTER 5

# CONCLUSION AND FUTURE WORK

## 5.1 Conclusion

Both the PSO and genetic algorithm for solving travelling salesman problem on average yield same effectiveness (solution quality) but particle swarm optimization is more computationally efficient (uses less number of function evaluations than genetic algorithm).

The result of particle swarm optimization for solving travelling salesman problem vary greatly by changing particle count (number of routes) as well as the maximum velocity consider in a implementation.

More the number of particles results in less number of iteration to get the target value. Although increasing the velocity using the implementation we have done have very little effect.

Genetic algorithm involves three operators selection, crossover and mutation, however it depends very much on the way the problem is encoded and which crossover and mutation methods used.

Particle swarm optimization is a better approach than genetic algorithm for solving travelling salesman problem.

## 5.2 Future Work

We will investigate other methods to compose velocities and heuristics under the PSO in future works. We will also explore the researches in variable velocities in future and we will also apply the proposed approach to the generalized TSP and to the Bi-objective TSP.

Work related to adding heuristics to genetic algorithm can be done in future to increase the probability of better result in less time.

# REFERENCES

1. Qinghai Bai. “*Analysis of Particle Swarm Optimization*”. College of Computer Science and Technology Inner Mongolia University for Nationalities. Computer and Information Science Vol. 3, No. 1, February 2010, pp.180-182.
2. Sapna Katiyar. “*A comparative study of Genetic Algorithm and the Particle Swarm Optimization”.* A.B.E.S. Institute of Technology NH-24, Vijay Nagar, Ghaziabad U.P. AKGEC International journal of Technology, Vol. 2, No. 2, 2004, pp.21-23.
3. Rania Hassan, Babak Cohanim and Olivier de Wrek. “*A comparison of particle swarm optimization and genetic algorithm”.* American Institute of Aeronautics and Astronautics, pp.4-5.
4. Chumming yang and Dan Simon. “*A new Particle Swarm Optimization Technique*”. Electrical and Computer Engineering Department. 18th International Conference on Systems Engineering 2005, pp.4-8.
5. Jaco F. Schutte. “*The Particle Swarm Optimization Algorithm*”. Structural Optimization Fall 2005, pp.4-8.
6. Dian Palupi Rini, Siti Mariyam Shamsuddin and Siti Sophiyati Yuhaniz. “*Particle Swarm Optmization: Technique, System and Challenges”.* International Journal of Computer Applications, Vol. 14, No. 1, January 2011, pp.19-22.
7. Anuj Sharma, Anshul Mehta. *“Observing the effect of elitism on the performance of GA”.* International Journal of Advanced Research in Computer Science and Software Engineering. Vol. 3, Issue 6, June 2013, pp.1474-1476.
8. Rajib Kumar Bhattacharya. *“Introduction to Genetic Algorithm”*. IIT Guwahati, November 2013, pp.5-20.
9. Scott M. Thede. “*An introduction to GA*”. De Pau University, Green Castle, pp. 1-6.
10. Darrell Whitley. *“A Genetic Algorithm Tutorial”,* pp. 1-8*.*
11. Zakir H. Ahmed. “*GA for the TSP using Sequential Constructive Crossover Operator*”, pp.5-7.
12. Melanie Mitchell. *“An Introduction to Genetic Algorithms”,* pp.3-6*.*
13. David S Johnson. *“Travelling Salesman Problem: A case study in local optimization”.* November 1995, pp. 3-8.
14. R.N. Mondal, S.K. Saha. “*An Approach for Solving Travelling Salesman Problem”*. International Journal of Applied Operational Research, Vol. 3, No. 2, Spring 2013, pp. 16-17*.*
15. H.P. Williams. *“The Travelling Salesman Problem”,* pp*. 2-10.*
16. Simon de Givry. *“A brief introduction to Traveling Salesman Problem”,* pp. 3-6*.*
17. Gilbert Laporte. *“The Travelling Salesman Problem: An overview of exact and approximate algorithms”.* European Journal of Operational Research, 1992, pp. 231-234.
18. Xuesong Yan Can Zhang, Wenjing Luo, Wei Li, Wei Chen. *“Solve Traveling Salesman Problem Using Particle Swarm Optimization Algorithm”*. IJCSI International Journal of Computer Science Issues, Vol. 9, Issue 6, No 2, November 2012, pp. 264-266*.*
19. Gajendra Singh Chandel, Ravindra Gupta, Arvinda Kushwaha. *“Implementation of Shortest Path in Packet Switching Network Using Genetic Algorithm”.* International Journal of Advance Research in Computer Science and Software Engineering, Volume 2, Issue 2, February 2012, pp .1-5.
20. Xuesong Yan, Qinghua Wu and Hanmin Lu *“An Improved Particle Swarm Optimization Algorithm and its Application”.* IJCSI International Journal of Computer Science Issues, Vol. 10, Issue 1, January 2013, pp. 1-5.
21. http://stackoverflow.com/questions/19755397/shortest-path-using-particle-swarm- optimization.
22. <http://www.lalena.com/AI/Tsp/>
23. <http://en.wikipedia.org/wiki/Travelling_salesman_problem>
24. en.wikipedia.org/wiki/Genetic\_algorithm/
25. en.wikipedia.org/wiki/Hamiltonian\_path