Global Journal of Pure and Applied Mathematics.
ISSN 0973-1768 Volume 13, Number 6 (2017), pp. 2009-2031
© Research India Publications
http://www.ripublication.com

Predicting optimal path for goods delivery to fair price shops in India using Hierarchical Intelligent Water Drops Algorithm

R.Sathish Kumar[1*], C.Rani[2] and P. GaneshKumar[3]

[1 2] Dept. of Computer Science and Engg, Government College of Engineering, Salem-11, Tamil Nadu, India.
[3] Dept. of Information Technology, Anna University Regional Campus, Coimbatore-46, Tamil Nadu, India.

Abstract

Designing a smart system for delivering goods to various fairprice shops effectively is one of the major goals in the mission of smart city development in India. In this paper, the smart city environment is treated as a distributed environment for carrying goods across different parts of the city. Intelligent water drops algorithm is implemented hierarchically as MapReduce model using hadoop environment to compute the optimized path for speedy delivery of goods. A front end is designed to carry out automatic request generated from the different fair price shops. Required items for each fair price shops is viewed using show requirement option in both district and taluk level. Distance matrix comprising of 100, 200, 300,…., 500 cities are generated using google maps and used for the simulation. From the experiment, it is observed that the proposed hierarchical intelligent water drops algorithm is implemented in MapReduce model meets the objective of delivering the ration materials to all the required fair price shops in minimum time. Further it is observed that, the path predicted by the proposed algorithm is found to be optimal among districts, taluks and fair price shops than other competitive optimization algorithms.

Keywords: Smart City, Smart Fair price shop, Intelligent Water Drops algorithm, MapReduce, Hadoop Environment.

1. INTRODUCTION

Public Distribution System (PDS) in India is established by the Government of India under Ministry of Consumer Affairs, Food and Public Distribution on June 1997 and it is jointly managed by the state governments in India. The fair price shop is a part of the Indian public distribution system which is established by the government of India to distribute items like wheat, rice, kerosene and sugar at a very less price to the poor people. Locally, this shop is called as “ration shops”. The item sold in this shop is much cheaper with good quality. This fair price shops are located in each districts, towns, and villages of each state. While the central government is responsible for the procurement, storage, transportation and bulk allocation of food grains and the state government holds the responsibility for distributing to the people through the fair price shops.

On September 2014 the honorable Prime Minster of India Narendra Modi has announced to develop 100 smartest cities in India under the Union Ministry of Urban Development. Smart City (SC) mission is to improve the interactivity between the citizens and the government. The SC mission is to enhance the interactivity of urban services between the citizens and the government and to reduce the resource consumption cost and facilitate to lead the better-quality life. The vision of SC’s is to improve the management, urban workflow and allow to response for the real time challenges. The aim is to collaborate with all respective state government of India to renovate the existing cities and towns in India. Among the five major elements to develop the smart cities in India, by providing the improved performance and to secure the Public Distributed System in a smarter way. It is a key element promoted widely by the urban development ministry of India. The idea behind the development of Smart Transportation System for Fair Price Shop (STS-FPS) discussed in this paper is to improve the performance of task to deliver the goods. This STS-FPS will improve the high quality of life through Information and Communication Technology (ICT) and allows fair price shop officials to work together directly with the society and to monitor the stocks that available in each fair price shops using the real time sensors.

The objective of the proposed work is to develop an algorithm to deliver the ration items to the required fair price shops automatically, once the request from the particular fair price shop is generated. The containers of food items in each fair price shops are attached with bin level indicators to indicate the level of ration items in fair price shop. These bin level indicators are used to detect the presence and absence of food material at the predetermined points within a container, once the ration material in the fair price shop goes to medium level, the bin level sensor will detect and send request to the central warehouse present in the particular district. In the central warehouse, once the request is raised, then the required items for the particular fair price shop is added to the database and the report is generated.

India is the second largest road network country in the world. Different states with different cities connect with the roads all over the India. But there is some unexplored path which is still not identified by the people during the transport of goods within

India. Further, safety and reliability are other issues needs to be taken care seriously during the transfer of goods from the source to point of delivery of goods. In general, there are many routes considered for transporting the goods. This work focus on providing a solution to such challenge task of identifying the optimal path based on heuristic algorithm namely Intelligent Water Drops (IWD) algorithm. This IWD algorithm is implemented hierarchically in district and taluk wise to recommend a path with minimum distance, time and effort that resulted in an effective transportation of goods to the fair price shop. The rest of the paper is organized as follows. In section 2, architecture of the proposed smart fair price shop transportation system in smart city development is given. The data generation for the smart fair price shop transportation system is discussed in section 3. In section 4 intelligent water drops algorithm is discussed. Hierarchical implementation of MapReduce intelligent water drops algorithm is discussed in section 5. Simulation result is given in section 6. Finally concluding remarks with the scope for the future extension are discussed in section 7.

2. SMART FAIR PRICE SHOP TRANSPORTATION SYSTEM

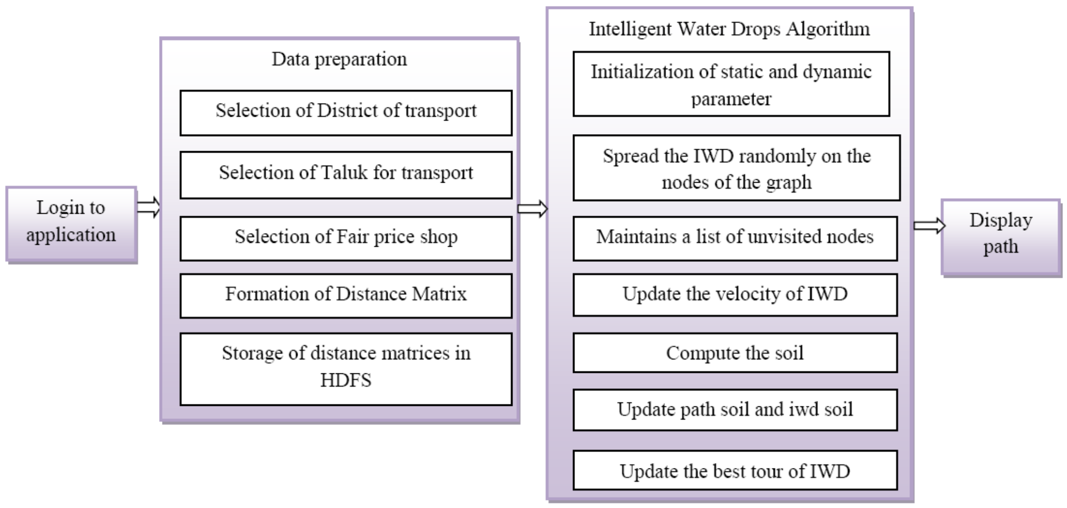

Login to application
Data preparation
Selection of District of transport
Selection of Taluk for transport
Selection of Fair price shop
Formation of Distance Matrix
Storage of distance matrices in HDFS
Intelligent Water Drops Algorithm
Initialization of static and dynamic parameter
Spread the IWD randomly on the nodes of the graph
Maintains a list of unvisited nodes
Update the velocity of IWD
Compute the soil
Update path soil and iwd soil
Update the best tour of IWD
Display path

Figure 1: Architecture of Smart transportation system for fair price shop

Data for the proposed smart fair price shop is generated from the bin level sensors that are fixed in the various containers of each fair price shops. The generated data from these sensors are stored separately by district and taluk level in the database using sensor grid (integrated technology of wireless sensor networks with grid computing) through internet. Based on the level indicated by the sensor, the requirement of items for each fair price shop is noted using a threshold limit and the same is made visible in the application suitably. From the application, name of the district, taluk and fair price shops that needs transportation of food items are chosen. The latitude and longitude distance of each district, taluk, fair price shop are taken from the google maps and it is stored in the form of adjacency matrix in the hadoop distributed file system. The objective of this work is to find the optimized path to deliver the food items to the needed district, within each district to the needed taluk, within each taluk to the needed fair price shops in hierarchical manner with minimum time. In order to achieve the goal, this paper suggests to implement the IWD algorithm hierarchically in hadoop distributed environment.

Intelligent water drops is an algorithm conceived by Shah-Hosseini in 2007 to solve various optimization problems based on the nature of water drops actions and the reactions that occur between the river bed and the water drops. The IWD algorithm includes a number of computation agents (i.e. water drops). After each iteration, the water drops constructs a solution, by using finite set of discrete movements. Every water drop starts with an initial state, and then it iteratively moves ahead until a complete solution is produced. Each water drop combining with other water drops to update the environmental properties like soil and velocity. At the beginning each water drop starts with an initial velocity and carries zero amount of soil. When the water drops moved from one place to another place, its soil level and velocity are updated. The velocity is changed non-linearly, and it is proportional to inverse the amount of soil between two locations. The water drop carries an amount of soil which is non-linearly proportional to the inverse of time taken by the water drop to move from current location to the next location. On the other hand, the time taken by the water drop to travel from one location to another location is proportional to its velocity and it is inversely proportional to the distance between two locations**.**

The Intelligent water drops algorithm is inspired by the natural flow of the river water that finds the optimal path from the action and reaction occurring among the water drops in the river beds. IWD optimization algorithm is developed using population based optimization techniques and the solution is constructed incrementally. The IWD algorithm is applied for a number of IWDs to find the optimal solution to the given problem. The problem of IWD is represented using graph (N, E) here N is the nodes and the E is the edges on the given graph. Each IWD begins to construct its solution by travelling between the nodes of the graph along with the edges until it completes its solution which is denoted by IWD T. Once all IWD complete their solution then it is considered as one iteration. After each iteration the best solution is found and is represented as IB T. The iteration based solution IB T is the best solution among all the solution which is obtained by the IWD. IB T is used to update the total best solution TB T. The total best solution TB T is the best solution of the IWD algorithm, which has been found among all iterations.

3. DATA GENERATION

The details of the various components needed for data generation for the proposed STS-FPS are discussed under the subsections below.

3.1 Sensor level data generation

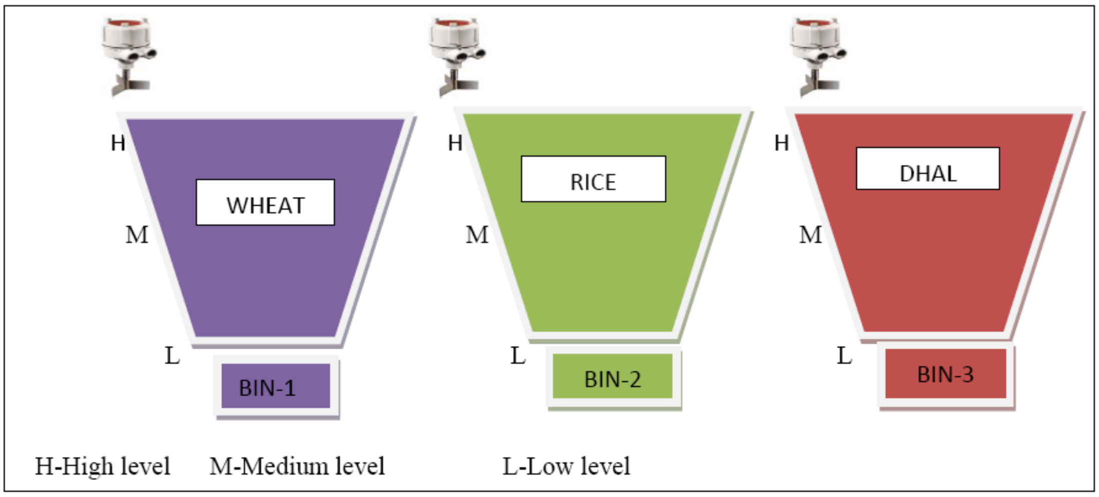


Figure 2: Bin level sensors that montior the container in fair price shop

The bin level sensor installed with each container in the fair price shop is shown in the figure 2.This bin level sensors monitor the ration material (rice, wheat, dhal etc.) placed in the container continuously and prevents the bin overflow by giving alert message. Further an important aspect of using the sensor is, once the level of the material in the bin goes below the medium level, the sensor will automatically alert and send the current stock details to the application available in the central warehouse of each district. The generated data from these sensors are stored in the database using sensor grid technology through internet.

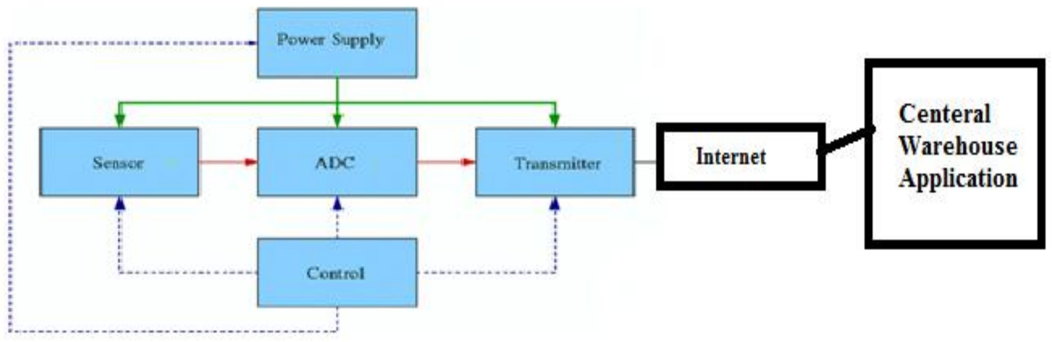
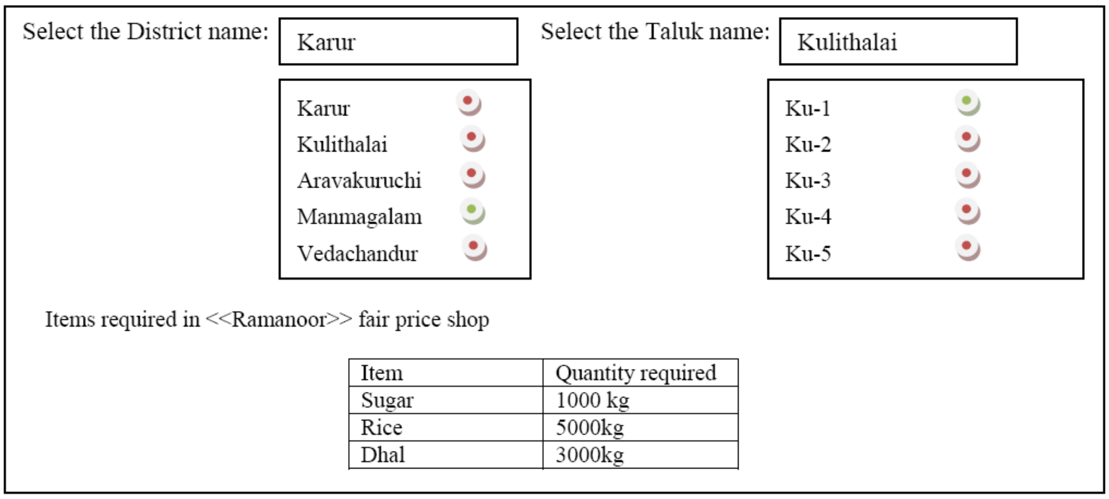


Figure 3: Data transfer using sensor via internet to central warehouse

Data sensed by the sensor will be in the form of analog. In order to process the data through internet application, digital data is required. For converting analog data into digital Analog to Digital Converter (ADC) conveter is used. The control unit and power supply unit are used to control the ADC converter for analog to digital conversion which is shown in figure 3.Finally the transmitter transfer the digital data through the transmitting medium (Internet) to the central warehouse present in each district's warehouse.

3.2 Application level data generation

An application is designed as shown in figure 4, to display the district name, taluk name and the items required in a fair price shop. At first, the user selects the district. Once the district is selected, the application will display the list of taluks present in that district with a light indicator as per sensed data. If the sensed data is below the medium level, red color light indicator is turned on. On the other hand, green color indicates there is no need for food material. The user will select the red color indicated taluk, to display the list of fair price shop. Each fair price shop is also listed using color indicator. Once the particular fair price shop in the taluk is selected then it automatically list out the items and their quantity required for the particular fair price shops.



Select the District name: Karur

Select the Taluk name: Kulithalai

Karur	🔴
Kulithalai	🔴
Aravakuruchi	🔴
Manmagalam	🟢
Vedachandur	🔴

Ku-1	🟢
Ku-2	🔴
Ku-3	🔴
Ku-4	🔴
Ku-5	🔴

Items required in <<Ramanoor>> fair price shop

Item	Quantity required
Sugar	1000 kg
Rice	5000kg
Dhal	3000kg

Figure 4: Application level data generation for proposed fair price shop

At this particular stage, the number of required district along with district name is sent to the data base for creating distance matrix to generate the optimal path district wise in the first level of IWD algorithm. After that, number of taluks along with their taluk name is sent to data base for creating distance matrix to generate optimal path taluk wise in the second level of IWD algorithm. Finally, number of fair price shops along with their fair price shop name are sent to data base for creating distance matrix to generate optimal path fair price shop wise in the third level of IWD algorithm.

3.3 Data base creation

The proposed STS-FPS application has four tables' namely District, Taluk's, Shop, and requirement. In the district table, the attributes used are district id, name of the district, latitude and longitude .In the taluk's table, the attributes are such as taluk id, district id, name, latitude, longitude and status of the requirement. In shop table the following attributes are used such as Shop_id, taluk_id, name, latitude, longitude, status of requirement, stock_rice, stock_wheat, stock oil, stock_sugar, stock_kerosene, allotted_rice, allotted_wheat, allotted_oil, allotted_sugar and allotted_kerosene are stored. The requirement table consists of following attributes such as district id, shop id, req rice, req wheat, req oil, req sugar, req kerosene are stored in this table. The relationship between the tables and their attributes are shown in the figure 5.

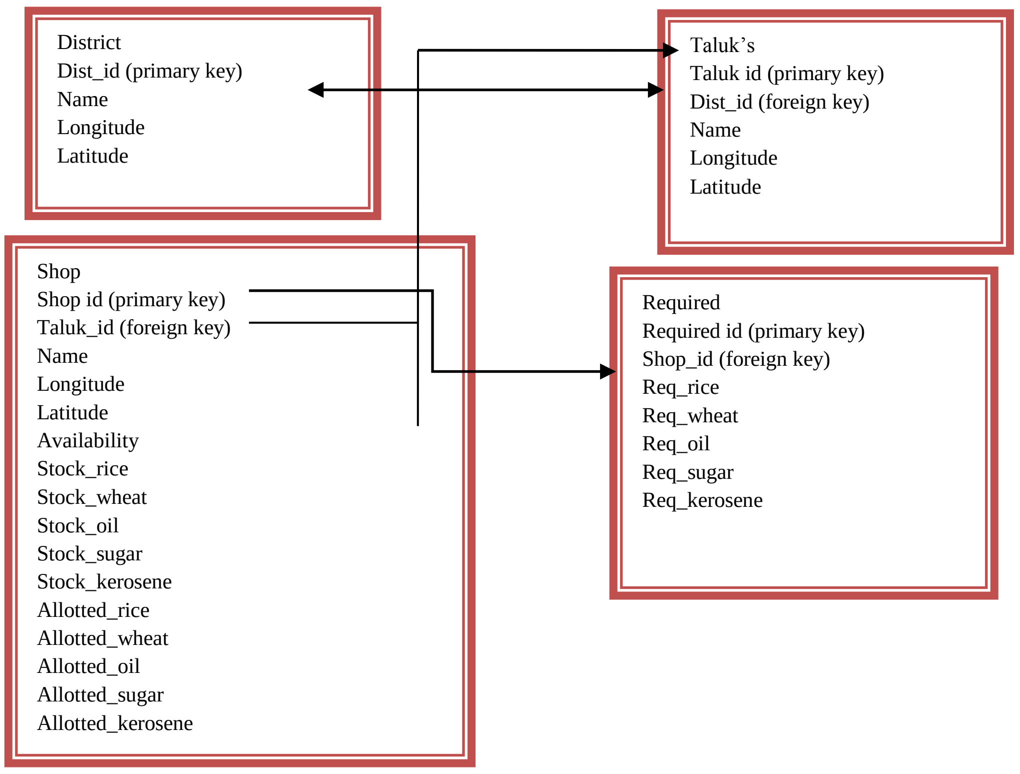
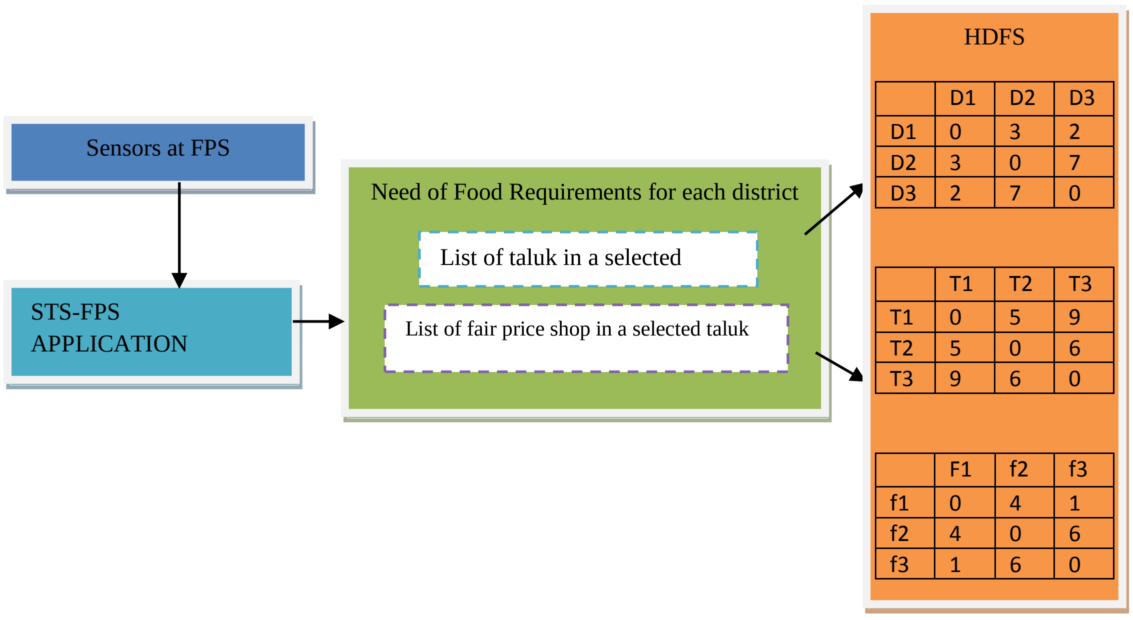


Figure 5: Ration ship diagram for the proposed STS-FPS application

All these four tables are constructed under **"Associative"** property. District table to Taluk table have one to many relationship (1-M), Taluk table to shop table have one to many relationship (1-M), and shop table to required table have one to many relationship (1-M), In order to reduce the replication of data attributes between the table's primary keys are used and for multiple reference between each tables foreign keys are used. The bin level sensor senses the level of food material in the container and updates the current stock of the particular fair price shop in shop table. Based on the purchasing plan table, the required amount of stock for each fair price shop is allotted and the allotted items and its quantity are also updated in the shop table.

3.4 Distance Matrix creation

Through STS-FPS application, based on red light indicator, the list of taluks and the list of fair price shop in the particular taluk's are selected for food item transportation. Using the google maps, the distances among each taluk and their fair price shops are formed in the form of distance matrix and stored in Hadoop Distributed File System (HDFS).


Sensors at FPS
STS-FPS APPLICATION
Need of Food Requirements for each district
List of taluk in a selected
List of fair price shop in a selected taluk
HDFS

	D1	D2	D3
D1	0	3	2
D2	3	0	7
D3	2	7	0

	T1	T2	T3
T1	0	5	9
T2	5	0	6
T3	9	6	0

	F1	f2	f3
f1	0	4	1
f2	4	0	6
f3	1	6	0

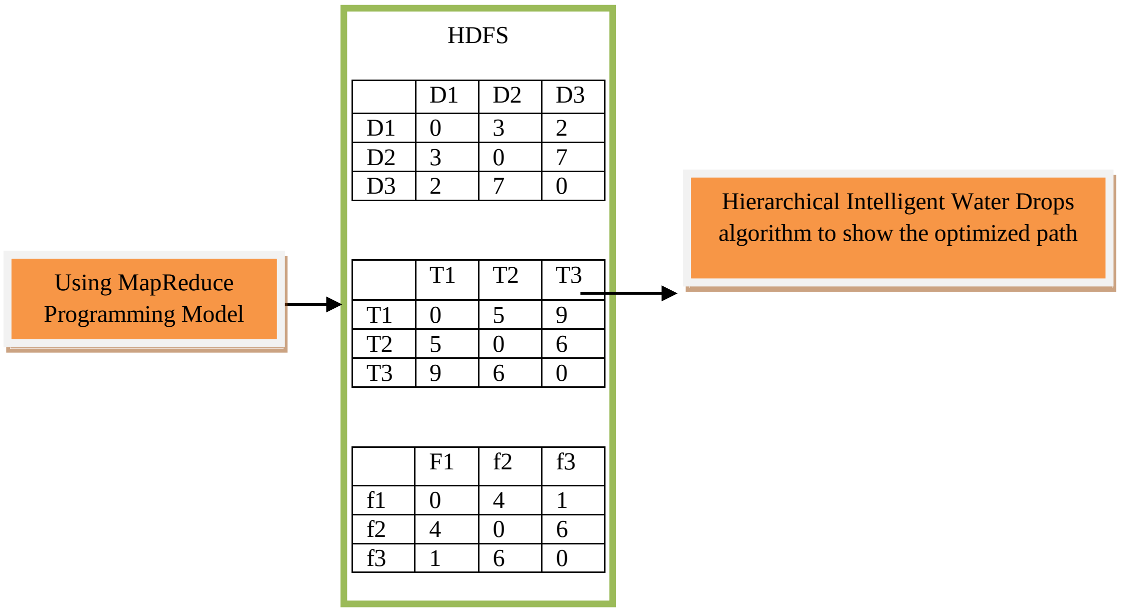
Figure 6: Distance Matrix creation

The district, taluks and fair price shop found within selected district are labeled as nodes. Then each node's connection is given a weight that represents the distance between the places. An adjacency matrix is generated as shown in figure 6 and it is stored as a distance matrix in a

hadoop distributed file system.

3.5 Hadoop Distributed file system storage

The distance matrices generated by the STS-FPS application are stored in the distributed manner. In order to process the data under distributed environment, the distributed storage is necessary. This storage is provided by the Hadoop environment namely Hadoop Distributed file system (HDFS).


HDFS
Using MapReduce Programming Model
Hierarchical Intelligent Water Drops algorithm to show the optimized path

	D1	D2	D3
D1	0	3	2
D2	3	0	7
D3	2	7	0

	T1	T2	T3
T1	0	5	9
T2	5	0	6
T3	9	6	0

	F1	f2	f3
f1	0	4	1
f2	4	0	6
f3	1	6	0

Figure 7: Hadoop distributed file system

In this regard, Hadoop framework with MapReduce program model will be an effective tool for processing the data in a distributed way to develop proposed HIWD algorithm. This work mainly focus on developing the nature inspired HIWD algorithm by solving in a distributed way by utilizing the MapReduce programming model as shown in figure 7 using Hadoop distributed file system.

4. INTELLIGENT WATER DROPS ALGORITHM

When solving the optimization problem using IWD algorithm, the first step is to represent the problem by using a fully connected weighted graph i.e., *G (V, E),* in this vertices are denoted by $V=\{V_i\,|i=1,.....N\}$,and arcs denoted by $E=((i,j)|(i,j)\in V\times V,\ i \neq j.i,j=1,.....N\}$.Here N is the number of decision variables. The feasible permutation of decision variable is represented by a finite set of components which includes vertices and arcs. $|| = \{a_k\,|\,K=1,....D\}$,where $D<N$ is the dimension of the solution, $a_k \in A$, here A is the set of all possible components which includes vertices and edges.

The first step is to represent the cities by nodes of a graph and the link in the graph represents the path joining each two cities. Each path has an amount of soil. An IWD can travel between cities through these links and can be change the amount of their soil. Therefore each node in the graph which holds the physical position of each city in the term of its two dimensional coordinates while the links of the graph denotes the path between the cities. To implement the limitation that each IWD never visits a city twice, we consider a visited city list for IWD which includes the cities visited by the IWD so for. So the selection of next cities for an IWD is to choose not be from the cities in the visited list. In the following, we present the intelligent water drops algorithm for the proposed SFP-STS. The intelligent water drops developed by Hamed Shah_Hosseini is given below [1]

1. At first step initialization of static parameters: set the number of water drops N_{IWD}, the number of cities N_c and the Cartesian coordinates of each city i such that $C(i)=[x_i \; y_i]^T$ to chosen constant value is carried out. The number of cities and their coordinates depend on the problem at hand while the N_{IWD} to equal to the number of cities N_c. For velocity updating we use parameters a_v=1, b_v=0.01, and c_v=1.For soil updating we use the parameter a_s=1, b_s= 0.01, and c_s=1.Moreover, the Initial soil on each link is denoted by the constant *InitSoil* such that the soil of the link between every two cities *i* and *j* is set by s*oil (i,j) = InitSoil.* The initial velocity of IWD$_s$ is denoted by the constant *InitVel.* Both parameters *IntiSoil* and *InitVel* are also user selected. In this paper we choose *InitSoil* =1000 and *InitVel*=100.The best tour is denoted by T_B which is still unknown and its length is initially set to infinity: $Len\ (T_{B)} = \infty$. Moreover we should specify the maximum number of iteration the algorithm should be repeated or some other terminating condition suitable for the problem.
2. Initialization of dynamic parameter: For every IWD, we create a visited city list
   $V_c\ (IWD)=\{\}$ set to the empty list. The velocity of each IWD is set to *IntiVel* whereas the initial soil of each IWD is set to zero.
3. For every IWD, randomly select a city and place the IWD on the city.
4. Update the visited city list of all IWD$_s$ to include the cities just visited.
5. For each IWD, choose the next city *j* to be visited by the IWD when it is in city *i* with the following probability

$$p_i^{IWD}(j) = \frac{f(soil(i,j))}{\sum_{k \notin vc(IWD)} f(soil(i,k))} \tag{1}$$

such that $f\big(soil(i,j)\big) = \frac{1}{\varepsilon_s + g(soil(i,j))}$

$$and \qquad g(soil(i,j) = \begin{cases} soil(i,j) \;\; if \min\limits_{l \notin vc(IWD)}(soil(i,j)) \geq 0 \\ soil(i,l) - \min\limits_{l \notin vc(IWD)}\big(soil(i,j)\big)\; else \end{cases}$$

Here $\in_s$ is a small positive number to prevent possible division by zero in the function *f ().*Here we use $\in_s$ =0.01.The function min () return the minimum value among all available values for its arguments. Most over *vc (IWD)* is the visited city of IWD.

6. For each moving from city *i* to city *j*, update its velocity as follows

$$vel^{IWD}(t+1) = vel^{IWD}(t) + \frac{a_v}{b_v + c_v.soil(i,j)} \tag{2}$$

such that *vel* ^{IWD} *(t+1)* is the updated velocity of the IWD. *Soil(i,j)* is the soil on the path joining the current city *i* and the new city *j.* using formula (2), the velocity of the IWD increase less if the soil is high the velocity would increase more if the soil is low on the path.

7. For each IWD compute the amount of the soil, Δ *soil(i,j)*,that the current water drop IWD loads from it's the current path between two cities *i* and *j*

$$\Delta soil(i,j) = \frac{a_s}{b_s + c_s.time\ (i,j;vel^{IWD})} \tag{3}$$

Such that $time(i,j;vel^{IWD}) = \frac{\|c(i)-c(j)\|}{\max(\varepsilon_v, vel^{IWD})}$

Computes the time taken to travel from city *i* to city *j* with the velocity vel^{IWD}. Here the function max () return the maximum value among its arguments, which is used here to threshold the negative velocities to a very small positive number $\in_v$=0.0001

8. For each IWD, update the soil of the path traversed by that iwd using the following formula

$$soil(i,j) = (1-\rho).soil(i,j) - \rho.\Delta soil(i,j) \tag{4}$$

$$soil(i,j) = soil^{IWD} + \Delta soil(i,j)$$

Where $soil^{IWD}$ represents the soil that the IWD carries. The IWD goes from city *i* to city *j.* The parameters ρ is a small positive number less than one .Here we use ρ =0.9

9. For each IWD, complete its tour by using step 4 to 8 repeatedly. Then, calculate the length of the tour *Tour* ^{IWD} traversed by the IWD, and find the tour with the minimum length among all IWD tours in this iteration. We denote this minimum tour by T_M

10. Update the soils of path include in the current minimum tour of the IWD, denoted by T_M

$$soil(i,j) = (1-\rho).soil + \rho.\frac{2.soil^{IWD}}{N_c(N_c-1)}\ \forall\ (i,j)\epsilon\ T_M \tag{5}$$

11. If the minimum tour T_M is shorter than the best tour found so far denoted by T_B, then we update the best tour by

$$T_B = T_M \text{ and } Len\ (T_B) = Len(T_M) \tag{6}$$

12. Go to step 2 unless the maximum number of iteration is reached or the defined termination condition is satisfied.

13. The algorithm stops here such that the best tour is kept in T_B and its length is $Len(T_B)$.

It is possible to use only T_M and remove step 11 of the IWD algorithm. However it is safer to keep the best tour T_B of all iteration than to count on only the minimum tour T_M of the last iteration

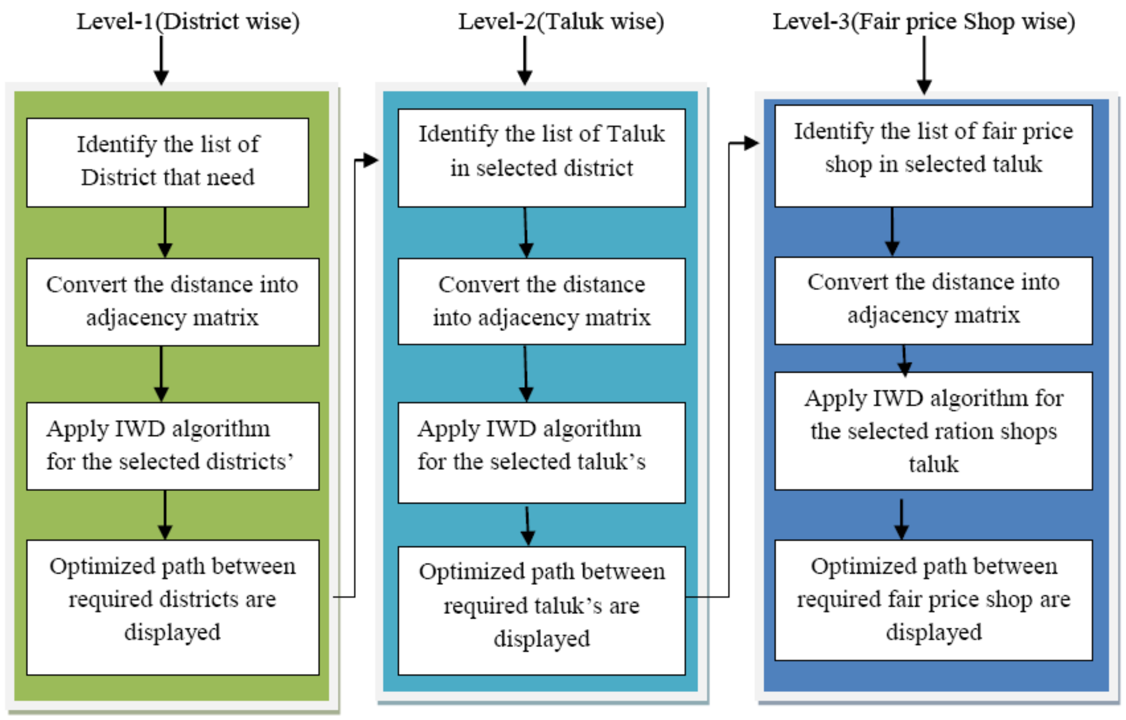
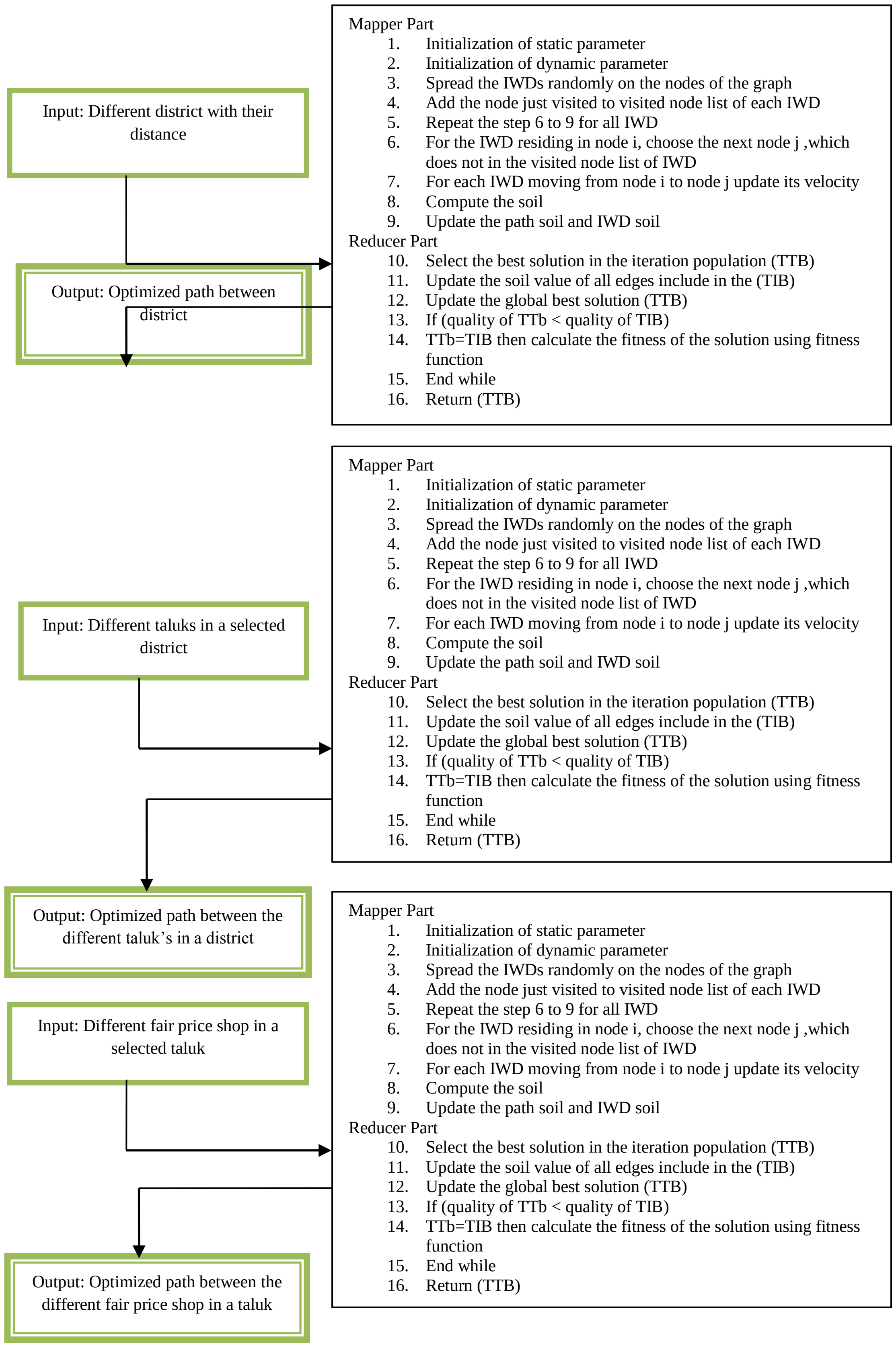

Level-1(District wise)
Identify the list of District that need
Convert the distance into adjacency matrix
Apply IWD algorithm for the selected districts'
Optimized path between required districts are displayed
Level-2(Taluk wise)
Identify the list of Taluk in selected district
Convert the distance into adjacency matrix
Apply IWD algorithm for the selected taluk's
Optimized path between required taluk's are displayed
Level-3(Fair price Shop wise)
Identify the list of fair price shop in selected taluk
Convert the distance into adjacency matrix
Apply IWD algorithm for the selected ration shops taluk
Optimized path between required fair price shop are displayed

Figure 8: Execution procedure of Hierarchical intelligent water drops algorithm

The proposed hierarchical intelligent water drops algorithm is executed under three different levels as shown in figure 8. In the first level, the name of the district and their distance are considered for execution to predict the optimized path district wise. In the second level, name of the taluk's and their distance of the selected taluk are considered for execution using IWD algorithm and the optimized path is achieved. In the third level, name of the fair price shops and their distance of the selected shops are considered to find the optimized path.

5. MAPREDUCE IMPLEMENTATION OF HIERARCHICAL IWD ALGORITHM

The proposed HIWD algorithm is implemented in Hadoop environment using MapReduce concept so that it will be better utilized for the proposed STS-FPS. The mapper and the reducer part of the proposed Hierarchical MapReduce Intelligent water drops algorithms at different levels are shown in figure below.



Input: Different district with their distance

Output: Optimized path between district

Mapper Part
1. Initialization of static parameter
2. Initialization of dynamic parameter
3. Spread the IWDs randomly on the nodes of the graph
4. Add the node just visited to visited node list of each IWD
5. Repeat the step 6 to 9 for all IWD
6. For the IWD residing in node i, choose the next node j ,which does not in the visited node list of IWD
7. For each IWD moving from node i to node j update its velocity
8. Compute the soil
9. Update the path soil and IWD soil

Reducer Part
10. Select the best solution in the iteration population (TTB)
11. Update the soil value of all edges include in the (TIB)
12. Update the global best solution (TTB)
13. If (quality of TTb < quality of TIB)
14. TTb=TIB then calculate the fitness of the solution using fitness function
15. End while
16. Return (TTB)

Input: Different taluks in a selected district

Output: Optimized path between the different taluk's in a district

Mapper Part
1. Initialization of static parameter
2. Initialization of dynamic parameter
3. Spread the IWDs randomly on the nodes of the graph
4. Add the node just visited to visited node list of each IWD
5. Repeat the step 6 to 9 for all IWD
6. For the IWD residing in node i, choose the next node j ,which does not in the visited node list of IWD
7. For each IWD moving from node i to node j update its velocity
8. Compute the soil
9. Update the path soil and IWD soil

Reducer Part
10. Select the best solution in the iteration population (TTB)
11. Update the soil value of all edges include in the (TIB)
12. Update the global best solution (TTB)
13. If (quality of TTb < quality of TIB)
14. TTb=TIB then calculate the fitness of the solution using fitness function
15. End while
16. Return (TTB)

Input: Different fair price shop in a selected taluk

Output: Optimized path between the different fair price shop in a taluk

Mapper Part
1. Initialization of static parameter
2. Initialization of dynamic parameter
3. Spread the IWDs randomly on the nodes of the graph
4. Add the node just visited to visited node list of each IWD
5. Repeat the step 6 to 9 for all IWD
6. For the IWD residing in node i, choose the next node j ,which does not in the visited node list of IWD
7. For each IWD moving from node i to node j update its velocity
8. Compute the soil
9. Update the path soil and IWD soil

Reducer Part
10. Select the best solution in the iteration population (TTB)
11. Update the soil value of all edges include in the (TIB)
12. Update the global best solution (TTB)
13. If (quality of TTb < quality of TIB)
14. TTb=TIB then calculate the fitness of the solution using fitness function
15. End while
16. Return (TTB)

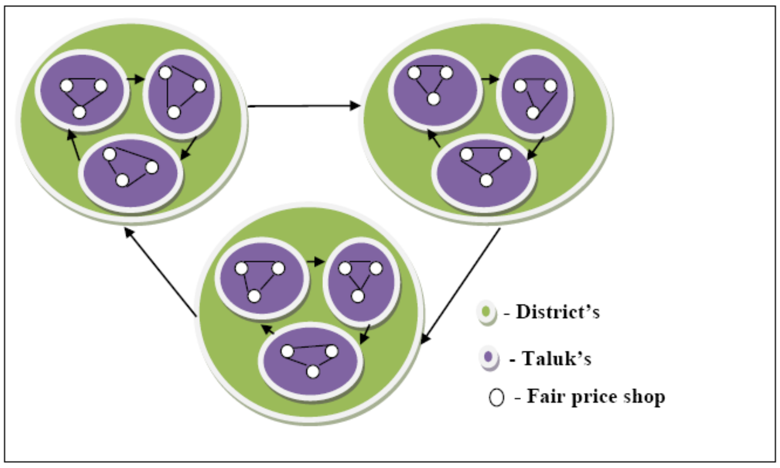


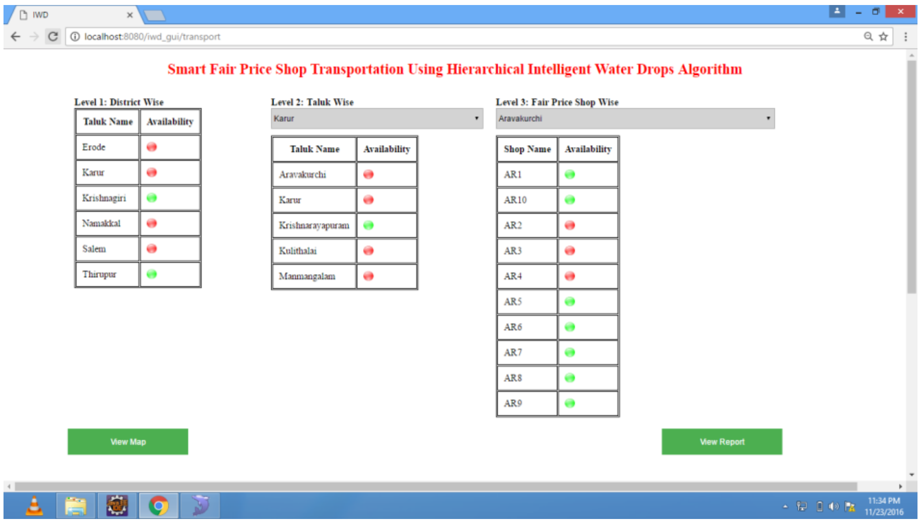
Figure 9: Path display under different level of HIWD algorithm

The different level of processing HIWD algorithm is shown in the figure 9. The green color shows the different district, the violet color represents the different taluk in the selected district and white color shows the different fair price shop in the selected taluks.

6. SIMULATION RESULT

6.1 User Interface design

In this section, the user interface designed for the proposed STS-FPS using Java SE 7 is discussed. The application development comprises of 4 modules namely 1. Requirements form 2.Database creation. 3. Report Generation and 4. Optimal path display. The developing tools and the scripting languages are namely Eclipse Java EE IDE, java server page, java script and SQlyog as database are used to develop the proposed STS-FPS web application.



Smart Fair Price Shop Transportation Using Hierarchical Intelligent Water Drops Algorithm

Level 1: District Wise

Taluk Name	Availability
Erode	🔴
Karur	🔴
Krishnagiri	🟢
Namakkal	🔴
Salem	🔴
Thirupur	🟢

Level 2: Taluk Wise

Karur

Taluk Name	Availability
Aravakurchi	🔴
Karur	🔴
Krishnarayapuram	🟢
Kulithalai	🔴
Manmangalam	🔴

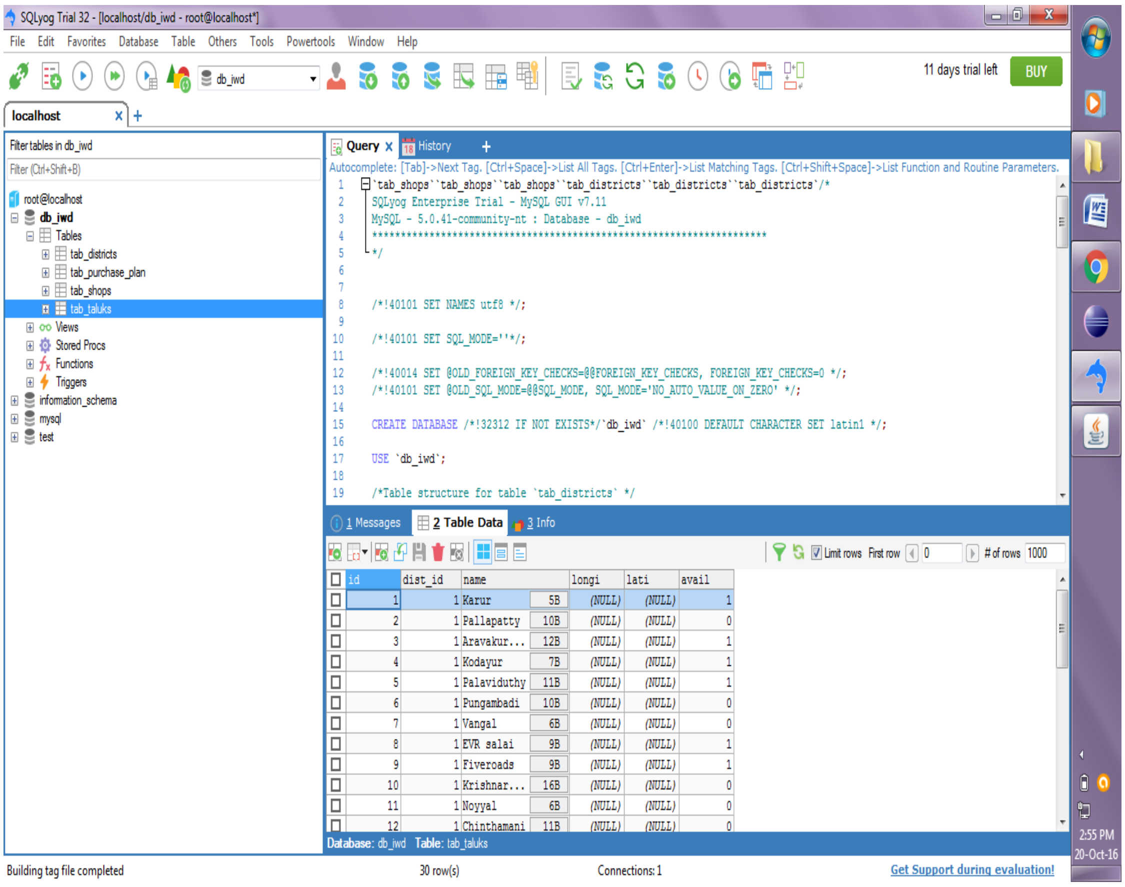
Level 3: Fair Price Shop Wise

Aravakurchi

Shop Name	Availability
AR1	🟢
AR10	🟢
AR2	🔴
AR3	🔴
AR4	🔴
AR5	🟢
AR6	🟢
AR7	🟢
AR8	🟢
AR9	🟢

Figure 11 GUI Design for Requirement form for Smart fair price shop

The Civil Supplies and Consumer Protection Department (CSCPD) in tamilnadu, is responsible for transmitting of ration goods to all fair price shop in and around tamilnadu. Information regarding the districts, taluks and fair price shops are collected from official website of CSCPD for building STS-FPS application. Figure 11 shows the Requirement form for transportation. This form includes input option like selection of districts, selection of Taluks, selection of fair price shops. Once the district is selected by the administrator, the corresponding taluk's for the selected district is displayed. In order to know the stock details in each taluk, an LED like indicator is fixed in the side of each taluk button. The green light indicates that there is no requirement needed for that taluk and the red light indicates that the requirements are needed. Similar design is carried out in fair price shops level also.



Autocomplete: [Tab]->Next Tag. [Ctrl+Space]->List All Tags. [Ctrl+Enter]->List Matching Tags. [Ctrl+Shift+Space]->List Function and Routine Parameters.

```
`tab_shops``tab_shops``tab_shops``tab_districts``tab_districts``tab_districts`/*
SQLyog Enterprise Trial - MySQL GUI v7.11
MySQL - 5.0.41-community-nt : Database - db_iwd
*********************************************************************
*/


/*!40101 SET NAMES utf8 */;

/*!40101 SET SQL_MODE=''*/;

/*!40014 SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0 */;
/*!40101 SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='NO_AUTO_VALUE_ON_ZERO' */;

CREATE DATABASE /*!32312 IF NOT EXISTS*/`db_iwd` /*!40100 DEFAULT CHARACTER SET latin1 */;

USE `db_iwd`;

/*Table structure for table `tab_districts` */
```

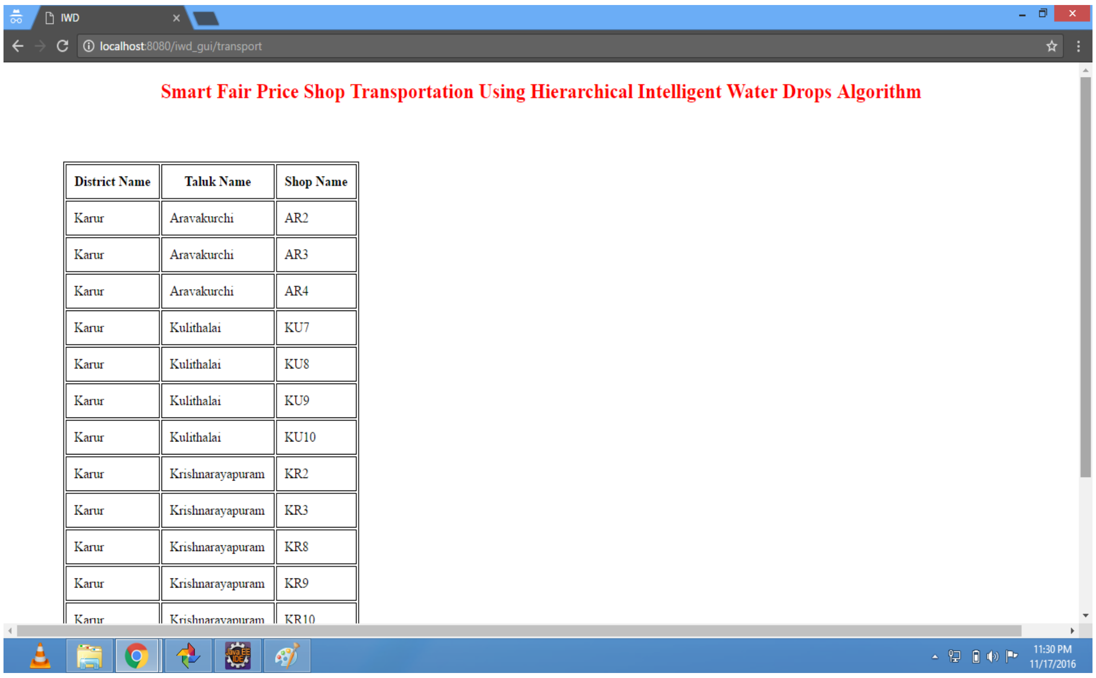
id	dist_id	name	longi	lati	avail
1	1	Karur 5B	(NULL)	(NULL)	1
2	1	Pallapatty 10B	(NULL)	(NULL)	0
3	1	Aravakur... 12B	(NULL)	(NULL)	1
4	1	Kodayur 7B	(NULL)	(NULL)	1
5	1	Palaviduthy 11B	(NULL)	(NULL)	1
6	1	Pungambadi 10B	(NULL)	(NULL)	0
7	1	Vangal 6B	(NULL)	(NULL)	0
8	1	EVR salai 9B	(NULL)	(NULL)	1
9	1	Fiveroads 9B	(NULL)	(NULL)	1
10	1	Krishnar... 16B	(NULL)	(NULL)	0
11	1	Noyyal 6B	(NULL)	(NULL)	0
12	1	Chinthamani 11B	(NULL)	(NULL)	0

Database: db_iwd Table: tab_taluks

Building tag file completed 30 row(s) Connections: 1 Get Support during evaluation!

Figure 12 Snapshot Database Creation form

Figure 12 shows the output of database creation module. In this module, the data filled by requirement forms are stored in the form of tables using SQlyog. Four tables are created to store the details of districts, taluks, fair price shop and lot of items.

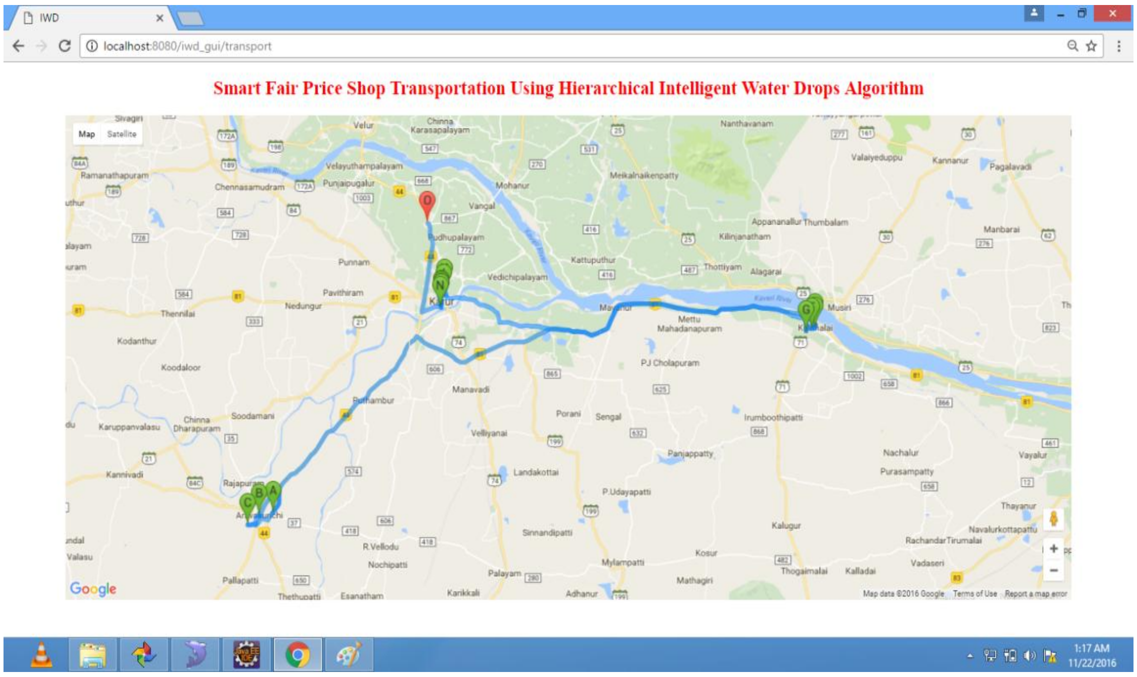


Smart Fair Price Shop Transportation Using Hierarchical Intelligent Water Drops Algorithm

District Name	Taluk Name	Shop Name
Karur	Aravakurchi	AR2
Karur	Aravakurchi	AR3
Karur	Aravakurchi	AR4
Karur	Kulithalai	KU7
Karur	Kulithalai	KU8
Karur	Kulithalai	KU9
Karur	Kulithalai	KU10
Karur	Krishnarayapuram	KR2
Karur	Krishnarayapuram	KR3
Karur	Krishnarayapuram	KR8
Karur	Krishnarayapuram	KR9
Karur	Krishnarayapuram	KR10

Figure 13 Details of Report Generation

Figure 13 shows the report generated form. The data stored in the database in generated in the form of a report which includes the District name, Taluk name, Fair price shop name.



Smart Fair Price Shop Transportation Using Hierarchical Intelligent Water Drops Algorithm

Figure 14 GUI Design for displaying the optimized path using HIWD algorithm

Figure 14 shows the optimized path using HIWD algorithm form. The requirement needed for particular district, taluk, fair price shop name are fetch from the database and these data are given to the algorithm and processed level by level and finally the optimized path is displayed.

6.2 Result

In this section the result of the proposed Hierarchical Intelligent Water Drops Algorithm (HIWD) for district level, taluk level, and fair price shop level are discussed. For the simplicity the performance of HIWD for karur district in tamilnadu is considered. In the second level, the HIWD is used to find the paths among applied taluk level in the karur district and its optimized path is shown in the table 1.The fair price shop present in different taluk in karur district and the requirement needed fairprice shops are highlighted and shown in Table 2.Finally required fairprice shops which are highlighted are selected for the third level of HIWD and the best optimized path is achieved with the minimum tour length, MapReduce processing time, java processing time are shown in table 3.

Table 1. MapReduce HIWD for selection of taluk in the district

District	**No. of Taluk's**	Taluk wise optimized path
Karur	5	Aravakuruchi->Kulithalai-> Krishnarayapuram->Karur->Manmagalam

Table 2. MapReduce HIWD for selection of path for fair price shop in taluk wise

District	**Name of the Taluk**	**Fair Price shop wise path**
Karur	Aravakuruchi	AR1,**AR2,AR3,AR4,**AR5,AR6,AR7,AR8,AR9,AR10
Karur	Kulithalai	KU1,KU2,KU3,KU4,KU5,KU6,**KU7,KU8,KU9,KU10**
Karur	Krishnarayapuram	KR1,**KR2,KR3,**KR4,KR5,KR6,KR7,**KR8,KR9,KR10**
Karur	Karur	KA1,KA2,KA3,KA4,KA5,KA6,KA7,KA8,**KA9,KA10**
Karur	Manmagalam	**MA1,MA2,MA3,MA4,MA5,**MA6,MA7,MA8,MA9,MA10

In table 2, the list of fair price shops are displayed in taluk wise in which the red color indicates the need of ration materials for the fair price shop and black color denotes no need of ration materials for the fair price shops.

Table 3. MapReduce HIWD for selection of best path for fairprice shop in all taluks in karur district

Best tour achieved for fair price shop level using HIWD	Tour Length in Km	MapReduce Timing (Ms)	Java Execution time (Ms)
KR2,KA9,MA5,AR2,KU7,KA10,AR3,KU8,KR9 AR4,KA10,KR3,KU9,KR8,KU10	27	1994	2370

In table 3, the best path for all fair price shops of all taluks in karur district is displayed. The total length covered by the path, MapReduce processing time, java processing time are reported. The district chosen for execution is not a city, so less number of taluks and fair price shops are considered. When big cities like Chennai, Coimbatore, Salem, and Madurai are considered, it might have more taluks and fair price shops. So in order to process more number of fair price shops in the city level, the number of fair price shop is extended. For discussion, the number of fair price shops from 100 to 500 and their tour length, MapReduce processing time, java processing time is reported in table 4.

Table 4. Number of Fair price shop and their length, MapReduce and java processing time

S.No	Number of Fairprice shops	Tour length (Km)	MapReduce Processing time(Ms)	Java Processing time(Ms)
1	100	9933	4347	6147
2	200	16932	5582	7082
3	300	45832	7737	9237
4	400	95721	9391	11913
5	500	228103	10459	13598

The HIWD algorithm is applied for different number of fair price shops and their tour length, MapReduce processing time and java processing time are reported in table-4. The proposed STS-FPS application is developed using distributed environment so the MapReduce timing are considered for execution and the processing time of MapReduce is lesser than java processing time which has more benefits for the proposed STS-FPS application using HIWD algorithm.

6.3 Comparison with Similar Optimization Algorithms

The Ant Colony Optimization (ACO) algorithm is one of the swarm intelligence metaheuristic algorithms, which are used for solving the computational problem by finding the shortest path using graphs. This algorithm works by the behaviour of moving ants from their nest to the food source using the shortest path. The moving ant deposit pheromone while travelling on the path so that ants coming behind will follow the same path instead of following the random path. Over time, pheromone start evaporates on the path so it states that, the particular path is no longer used by the other ants. The density of the pheromone plays vital role in choosing shortest path for the ants for achieving the global optimum solution of the algorithm. The overall result of ant colony optimization algorithm states that once the ant finds an optimized path from the colony to a food source then other ant will follow the same path.

The parameters to test ACO algorithms are fixed by the user, namely m - number of ants, α - pheromone relatively importance, β - relative importance of heuristic factor. ρ- Pheromone evaporation coefficient ((1-ρ) indicating the pheromone persistence factor).Q - amount of pheromone ants release. The parameters are m=500 α=1 β=3 ρ=0.5 q=10 number of iteration of the algorithm is 200. The ACO algorithm is executed for different number of fair price shops and the optimum length, MapReduce processing time, java processing time is recorded and displayed in table 5.The parameters to test HIWD algorithms are the Velocity and Soil updating parameters are fixed by the user such as for the velocity $A_{v=}1$, B_v=0.01, C_v=1, A_S=1, B_s=0.01, C_s=1,initial velocity of the soil is 10000, initial velocity =200, P_{iwd}=0.9, P_n=0.9, Maximum number of iteration for each IWD is 1000 and total execution of the algorithm is set to 200 iteration. The HIWD algorithm is executed for different number of fair price shops and the optimum length, MapReduce processing time java processing time is recorded and displayed in table 6. The comparison between ACO and HIWD algorithms are done on the similar property of algorithm called changing of values. The pheromone deposited on the travelling path in ant colony algorithm and amount of soil and velocity changed by the water drop in the travelling path are considered. The global optimum solution achieved by HIWD algorithm is better than ACO algorithm. The HIWD and ACO algorithm executed on hadoop single node cluster using 64 bit Ubuntu operating system with 4GB of RAM.

Table 5. Comparison of HIWD and ACO Optimization algorithm for different fair price shops

S.No	**Number of fair price shops**	**Tour length (Km)**		**MapReduce Processing time(Ms)**		**Java Processing time(Ms)**	
		HIWD	ACO	HIWD	ACO	HIWD	ACO
1	100	**9933**	10124	**4347**	4348	**4932**	4933
2	200	**16932**	17940	**5582**	5584	**6021**	6023

3	300	**45832**	46541	**7737**	7739	**7999**	8001
4	400	**95721**	96729	**9391**	9393	**9945**	9947
5	500	**228103**	234212	**10459**	10459	**10953**	10953

Table 6. Comparison of different HIWD and ACO for selected fair price shops in karur district

Best tour achieved for selected fair price shop using different algorithms	**Tour Length in Km**	**MapReduce Timing (Ms)**	**Java Execution Algorithms (Ms)**	**Algorithm**
KR2,KA9,MA5,AR2,KU7,KA10,AR3,KU8,KR9,AR4,KA10,KR3,KU9,KR8,KU10	27	1994	2270	HIWD
AR4,KA10,KR3,KU9,KR8,KU10KR2,KU7,KA10,AR3,KU8,KA9,MA5,AR2,KR9,	29	1996	2274	ACO

The comparison result of HIWD algorithm and ACO algorithms with their tour length, MapReduce processing time for the selected fair price shop in karur district is shown in table 6. In the point of execution, it is proved that HIWD algorithm take minimum length, execution time than ACO algorithm and gives the best optimization path for travelling. The objective of this work is to distribute the ration goods from to all fairprice shops in optimized path that is meet out by the developed HIWD algorithm.

6.4 Wilcoxon signed rank test

Wilcoxon signed rank test is a non parametric statistical hypothesis test used for comparing two related samples to assess whether their population means rank differ. The logic behind the Wilcoxon test is very simple. [16] The datas are ranked to produce two rank totals, one for each condition. If there is a systematic difference between the two conditions, then most of the high rank will belong to one condition and most of the low rank will belong to other one. As a result, the rank total will be quite different and one of the rank totals will be quite small. On the other hand, if the given two conditions are similar, then high and low rank will be distributed evenly between two conditions and rank total will be fairly similar and quite large. The Wilcoxon test statistic "W" is simply the smaller of the rank totals. The smaller it is (taking into account depending on how many participants) then the less likely it is to have occurred by chance. A table of critical values of W show how likely it is to obtain your particular value of W purely by chance. Note that Wilcoxon test is unusual in this aspect, normally, the bigger test statistic, and the less likely it is to have occurred by chance. The table 8 shows the Wilcoxon singed rank test is applied for HIWD and ACO algorithm based on their MapReduce processing time.

Table 7. Wilcoxson singed rank test for comparing HIWD and ACO algorithm

Fair price shops	HIWD	ACO	Difference	Rank
50	2153	2155	-2	5.5
100	4347	4348	-1	2
150	4992	4991	1	2
200	5582	5584	-2	5.5
250	6356	6357	1	2
300	7737	7739	-2	5.5
350	8463	8467	-4	8.5
400	9391	9393	-4	8.5
450	9896	9898	-2	5.5
500	10459	10459	0	0

According to the Wilcoxon signed rank test two hypotheses have been selected as H0 and H1. H0- denotes the significance difference is occurred between two algorithms and H1- denotes the significance difference is not occurred between two algorithms. Positive sign rank are added (2+2) = 4, Negative sign rank are added (5.5+2+5.5+5.5+8.5+8.5+5.5) = 41 and N=10 (N-1) =9.The critical value W=4 use the table of critical Wilcoxon values with the N of 9, the critical value for a two tailed test at 0.05 significance level is 6. The obtained value of 4 is lesser than critical value of 9 (4>6) from the observation from table 7 we can conclude that there is significant difference between HIWD and ACO algorithm and H0 hypothesis is satisfied. The outcome of the HIWD is found to be better optimum than ACO algorithm for the proposed STS-FPS application.

6.5 Convergence of HIWD vs ACO algorithm

The performance of the proposed HIWD and ACO algorithm are tested by several runs and however it is not always guaranteed that the global optimum tour is found in initial run of the algorithms. After several runs of these two algorithms global optimum tour is achieved. The convergence analysis of HIWD and ACO is shown in figure 16.The maximum iterations for both HIWD and ACO algorithm is set to 200 and the tour length is keep on changing for every iterations from 76th iteration the tour length remains constant for remaining all iterations for proposed HIWD algorithm and from 94th iteration the tour length remains constant for remaining all iterations for ACO algorithm.

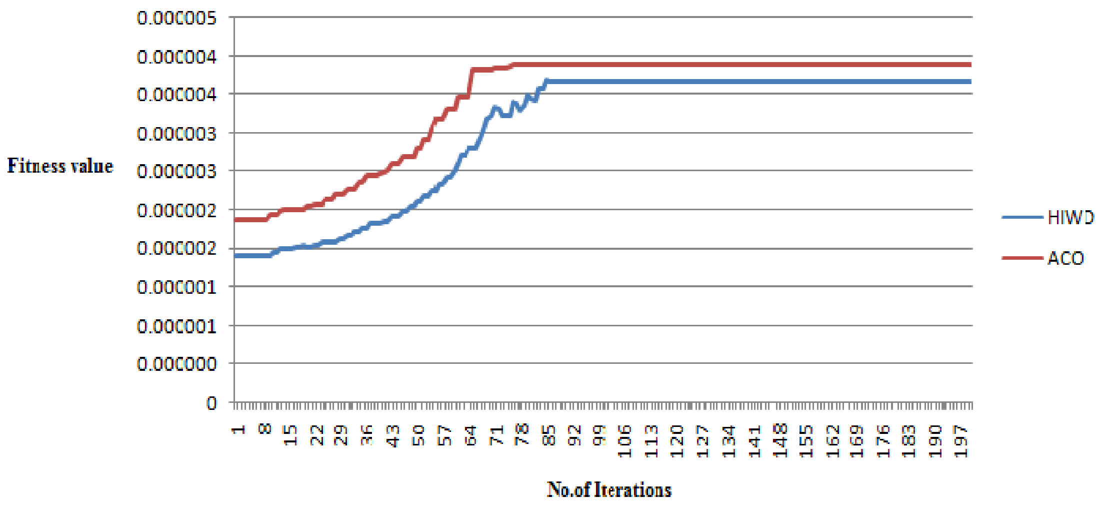


Figure16: Convergence comparisons between ACO and HIWD optimization algorithm

In order to calculation the fitness of the obtained solution it must be evaluated through fitness function namely $f(x) = 1/x$ here x denotes the distance. The proposed HIWD algorithm and ACO algorithm is made run on hadoop single node cluster using 64 bit Ubuntu operating system with 4GB of RAM. During the run it is noticed that, HIWD gives the minimum tour length than ACO algorithm which is show in figure 16.

7. CONCLUSION AND FUTURE ENHANCEMENT

An application is designed to compute the optimized path for STS-FPS. The HIWD and ACO algorithm are implemented using the Hadoop distributed framework. The distance between the districts, taluks and fair price shops are taken from the google maps and stored in the Hadoop distributed file system. Once the source and destination are given the algorithm automatically calculate the optimized path form the source to destination node. This kind of computation is more useful for the Indian civil supplies corporation (ICSC) to carry the food material within India with less time and with less fuel. In future the solution for loading the goods into the container is done in the optimized way and which more required by the ICSC. Heuristics or bio inspired optimization algorithms are planned to develop for finding a solution using Knapsack problem for Smart logistics transportation system in Hadoop environment.

REFERENCES

[1] Shan- Hosseini. H, “**Problem solving by intelligent water drops**” In IEEE 1088 Congress on evolutionary computing, DOI:10.1109/CEC.2007.

[2] Basem O. Alijla ,Li-Pei Wonga, Chee Peng Lime, Ahamad Tajudin Khadera, Mohammed Azmi AI-Beta "**An ensemble of intelligent water drops algorithm and its application to optimization problem**" Information Science pp. 175-189, 2015.

[3] Shah-Hosseini. H. **"Optimization with the nature –inspired intelligent water drops algorithm"** In Tech, pp. 297-320, 2009.

[4] Shah-Hosseini. H. **"Intelligent water drops algorithm for automatic multilevel thresholding of grey-level images"** Identification and control, pp. 241-249, 2012a.

[5] Shah-Hosseini. H. **"An approach to continuous optimization by the intelligent water drops algorithm",** Procedia-social and Behavioral Science, pp. 224-229, 2012b.

[6] Afaq, H., & Saini, S**."On the solution to the travelling salesman problem using nature inspired computing techniques"** International Journal of Computer Science, pp. 326-334,Vol. 8, 2011.

[7] Shan-Hosseini, H **"An approach to continuous optimization by the intelligent water drops algorithm",** Procedia-Soc.Behav.Sci ISSN. 1877-0428 pp. 224-229, 2012.

[8] Wang, X, Xu, G" **Hybrid differential evolution algorithm for travelling sales man problem"** Procedia Eng. 2011.

[9] Mo, H., Xu, L" **Research of biogeography swarm optimization for robot path planning**" Neuro computing, 2015.

[10] Shan-Hosseini, H **"Improving K-means clustering algorithm with the intelligent water drops (IWD) algorithm ",** Int.J.DataMin Model. Manag, pp.301-317, 2013.

[11] Yang, X-S **"A new met heuristic bat-inspired algorithm",** Nature Inspired cooperative strategies for optimization. pp. 65-74 2010.

[12] Ranieri Baragilla, Jose, Hidalgo, RAffaele Perego **"A parallel Hybrid Heuristic for the TSP"** Proceedings of 1st European workshop on Evolutionary Computing in Combinatorial Optimization. Pp.193-202. 2001.

[13] Jeffrey Dean, Sanjay Ghemawat. MapReduce**" Simplified Data Processing on Large Cluster"** Sixth Symposium on operating system Design and Implementation (OSDI), 2014.

[14] Claude N. Fiechter. **"A parallel tabu search algorithm for large travelling salesman problem"** Discrete Applied mathematics, Vol. 51, Issue. 3, pp. 243-267, 1994.

[15] John Knox "**Tabu search performance on the symmetric travelling sales man problem"** Computer and Operations Research, Vol. 21, issue. 8, pp. 867-876, 1994.

[16] **Graham hole research skills** version 1.0,Wilcoxenhandout 2011