Multi Ojective Optimized Travel Planner

Sonika Agrawal (UG201210035)
Divya Nagar (UG201213005)
Shreshtha Garg (UG201213035)

I. Abstract

Planning a travel is a very cumbersome process and takes a lot of time. From route of travel to accommodation ,there are many difficulties that come while planning. In our background research, we came to know about different travel planner sites available, which handle either accommodation, food, places or travel tickets at a time, but none, which take care of planning the route of travel. Route of travel is one of the most important factor in travelling. And this served as a motivation to take up the problem of finding the optimal route of travel.
The problem can be looked as an optimization problem with multiple objectives.There are different methods to solve optimization problems with different levels of objectives. According to our current approach initially we will try single objective multi constraint algorithms. In further steps we will start converting each constraint into objective and again solve the problem using Bi-objective and multiobjective optimizations algorithms. We are starting with the approach named as Goal Programming (Explained in detail in later part of document) which is a priori approach and later will perform an evolutionary approach.

II. Introduction

A. Overview and Problem Statement

As discussed above, we need to find a optimized travel route. The problem can be well understood with the help of a graph.

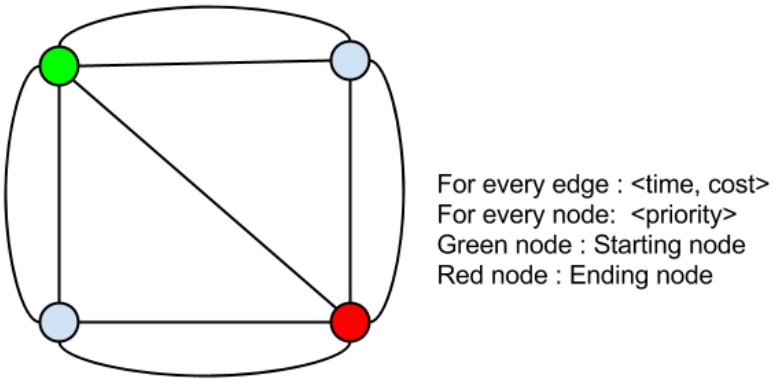

Fig. 1. Example graph

From the figure 1, there can 1, 0 or many edges between any 2 nodes. Every edge signify a possible mode of transport between the two nodes, having 2 parameters cost and time it takes to travel from one node to another. And every node has 1 parameter, that is the priority of the node. The node in green is the starting node (S), and node in the red is the ending node (E).

We wish to find a possible route from starting node to the ending node, such that the cost and time both are minimized and satisfy a given threshold. There is one more objective to take into account, that is, number of nodes the route should cover be maximized. Hence we are left with 3 objectives.

1. Minimizing Cost - Threshold Wc
2. Minimizing Time - Threshold Wt
3. Maximizing Number of places - Threshold Wn

Hence the problem statement become, finding a complete travel route optimizing the above written 3 objectives.

Complexity : If we consider only distance constraint, the problem get reduced to the famous ” The travelling salesman problem ” which tries to find, what is shortest possible route that visits every city once, starting from starting city and reaching at the destination city at the end. This Problem is a NP-Hard problem. Now coming to our original problem, which has 3 constraints, and many possible paths from one node to another node, also lies in the set of NP-Hard problems as, this problem can also be reduced to the above problem.

Hence with respect to the state of art, this problem lies in the hard problems, i.e. it is a NP-Hard Problem.

B. Motivation

In the current situation every traveller faces lots of problems while travelling, Planning a trip is very time consuming. All the data is available on the web but it is so scattered that it takes lots of time to get the correct and trustable results, Even if somehow someone has the data it requires too much efforts and time to design the plan in most optimized way with satisfying all the constraints. So the main motivation behind the project was to solve the problem of a common man who wants to travel/visit some places with lots of constraints and less knowledge about that place. The aim of technology is to make life easy and to make systems intelligent enough to take decisions or we can say to provide an overview of best possible solutions. We also have the same motivation behind this project, to make life easy.This kind of system can solve all these problems related to travelling which everyone faces in their life and this is why this system is important. Everyone who will use this system/product will get benefited.

III. Contributions of this work

Many problems from real world requires optimal route finding as a sub routine. We have proposed a solution method for optimal route finding by using different variants of goal programming. This work will help in analyzing the different solution approaches for solving the optimization problems.

The derived goal programming method is efficient as compare to other goal programming methods. It uses less number of variables as compare to the big-M method which we use for solving ”greater than” linear programming problems.

IV. Plan for future work

1. To implement the obtained solution methods.
2. To test other APIs like tixik,yelp to get better solutions for data collection.
3. To test the methods on the data collected and perform an analysis of different approaches.
4. To create a graphical user interface

V. Brief description of Work Done

The work done can be divided into two part:
1. Algorithm formation
2. Data collection

A. Algorithm formation

For formation of algorithm, we first found mathematical model of single entry single exit system and then general system. After that we derived a suitable goal programming model i.e weighted goal programming model and applied that on our mathematical models. There can be many ways to solve weighted programming model but we are considering two methods, 1. using dual and 2. using game theory. Further subsections will give details about each part of work.

1) Converting multi-edges graph into single edges graph:

From the problem statement , we have a graph containing multiple edges between any two nodes [Fig 2].Before we start solving the problem we have to convert this multi-edge graph into a graph, containing either 0 or 1 edge between any two nodes.

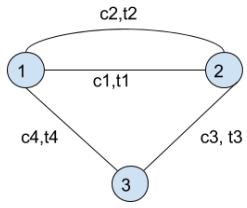

Fig. 2. Example graph

Method for conversion : Let’s suppose total number of nodes in the graph is n. If there are k edges between node i and node j, make (k-1) copies of node i and node j, numbering them p*n+i and p*n+j respectively where p = 1,2,.....,k-1. Make a edge between node p*n+i and node i, and node p*n+j and node j with cost and time 0, 0 respectively for p = 1,2,...,k-1. Also make a edge between node p*n+i to p*n+j node where, p = 1,2,....,k-1.We have created k-1 more edges. Now, we can delete k-1

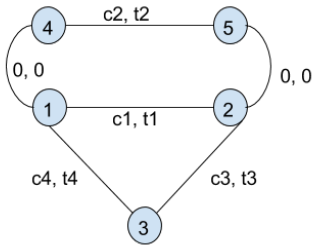

Fig. 3. Converted single edge graph for multi-edge graph in fig 2

edges between node i and j and assign their cost and time to newly created k-1 edges between the copied nodes.[fig 3] shows the converted graph for [fig 2].

As there can be at max 3 edges between any two nodes, the size of graph will not increase more than 3 times after doing the conversion.

2) Mathematical Formulation of single entry single exit system:
This section describes the mathematical formulation of all the underlying constraints of the problem statement.

a) Notatoins:
n = Total number of all nodes
C_{ij} = cost associated with edge from node i to node j
T_{ij} = time associated with edge from node i to node j
C_T = Maximum cost allowed
T_T = Maximum time allowed
M_{min} =minimum number of nodes to visit
Y_{ij} = inclusion variable associated with edge from node i to node j
Y_{ij} = 0 if edge from i to j is not included in final route, 1 otherwise

b) Constraints:
Constraint 1. Total travel time should be less than a given threshold value
Constraint 2. Total travel cost should be less than a given threshold value
Constraint 3. Total number of places visited should be more than a threshold value

Additional Constraints to ensure a continuous path and single entry and single exit :
Constraint 4: For every node, number of entrances should not exceed 1
Constraint 5: For every node, number of exits should not exceed 1
Constraint 6: For every node, number of entrances and exits should be equal

c) Mathematical formulation for all the constraints:
Constraint 1:

$$\sum_{i=1}^{n} \sum_{j=1 i \neq j}^{n} T_{ij} * Y_{ij} \leq T_T \tag{1}$$

Constraint 2:

$$\sum_{i=1}^{n} \sum_{j=1 i \neq j}^{n} C_{ij} * Y_{ij} \leq C_T \tag{2}$$

Constraint 3: As we are dealing with single entry, single exit system, Total number of visited nodes will be equal to total number of visited edges.

$$\sum_{i=1}^{n} \sum_{j=1 i \neq j}^{n} Y_{ij} \geq M_{min} \tag{3}$$

Constraint 4:

$$\forall i \in (1, 2, ..., n) \sum_{j=1 j \neq i}^{n} Y_{ij} \leq 1 \tag{4}$$

Constraint 5:

$$\forall j \in (1,2,...,n) \sum_{i=1 i\neq j}^{n} Y_{ij} \leq 1 \tag{5}$$

Constraint 6:

$$\forall i \in (1,2,...,n) \sum_{j=1 i\neq j}^{n} Y_{ij} - \sum_{k=1 k\neq i}^{n} Y_{ki} = 0 \tag{6}$$

3) Mathematical model of general problem statement:
For modelling general scenario from single entry single exit system, we have used X_is inclusion variable associated with nodes.

X_i = inclusion variable associated with node i
$X_i = 0$ if node i is not included in final route, 1 otherwise

And now we have n more equations that establish relation among X_is and Y_{ij}s and that is:
Logical equation for X_i :

$$X_i = Y_{i1}||Y_{i2}||.....||Y_{in} \quad \forall i \in (1,2,3...n) : \tag{7}$$

We can write the logical equation mathematically as:

$$X_i = \sum_{j=1 i\neq j}^{n} Y_{ij} - \sum_{j=1}^{n} \sum_{k=1 j\neq k}^{n} Y_{ij} * Y_{ik} + \sum_{j=1}^{n} \sum_{k=1 j\neq k}^{n} \sum_{l=1 l\neq j l\neq k}^{n} Y_{ij} * Y_{ik} * Y_{il} - ..... \tag{8}$$

This is a non-linear equation of order n. We can solve the mathematical model for general problem statement using the weighted goal programming model but can not use below described methods to evaluate weights, Hence we used preemptive goal programming model to solve it.

a) Preemptive goal programming model:
Preemptive goal programming allows us to solve linear goal programming problem without the headache of weights and also takes care of priorities. Only required information from decision makers for this approach is priorities of objectives. If a level has more than one goal then these goals can be solved using same non preemptive goal programming.

The idea behind preemptive goal programming model is that lower priority goals should not be attained at the expense of higher priority goals - They are preempted. That is, if total weighted deviations of priority 1 goals is V1, then, when seeking to minimize the total weighted deviations of priority 2, one must ensure that priority 1 value stays at V1. If the weighted deviations of priority 2 goals under this condition is V2, then, when seeking to minimize the weighted deviations of priority 3 goals, priority 1 objective value must remain V1 and priority 2 objective value must remain V2 and so on. So we compromise with the least priority goals and gives optimal values for higher priority goals.

4) Derivation of an efficient Goal programming model:
Goal Programming is a branch of multi-objective optimization. It is one of the oldest multi-criteria decision making techniques in which we try to minimize the deviation for each of our objectives from the desired targets. General goal programming problem formulation:

F = Set of feasible values which is defined by constraints of equalities and inequalities
$f_i(x)$ = Linear function of ith goal
g_i = Aspirational level of ith goal

Model-1

$$min \sum_{i=1}^{n} f_i(x) - g_i \tag{9}$$

$$S.T. x \in F, x \geq 0, g_i \geq 0 \tag{10}$$

To solve this problem with goal programming we assign deviations to our goals and then try to minimize the deviations. After putting deviations:
d_i^+ = Positive deviation from aspiration level for goal i
d_i^- = Negative deviation from aspiration level for goal i

$$\sum_{i=1}^{n} f_i(x) - g_i = d_i^+ - d_i^- \tag{11}$$

$$S.T. d_i^+ \geq 0, d_i^- \geq 0 \tag{12}$$

Now we can rewrite Model-1 as following:

Model-2

$$min \sum_{i=1}^{n} d_i^+ + d_i^- \tag{13}$$

$$S.T. f_i(x) - g_i - d_i^+ + d + i^- = 0 \quad \forall i \in (1, 2, ..., n) \tag{14}$$

$$d_i^+ \geq 0, d_i^- \geq 0 \quad \forall i \in (1, 2, ..., n) \tag{15}$$

$$x \in F, x \geq 0, g_i \geq 0 \tag{16}$$

By observing constraint of problem-2 we can write:

$$d_i^- = -f_i + g_i + d_i^+ \quad \forall i \in (1, 2, ..., n) \tag{17}$$

$$d_i^+ + d_i^- = 2d_i^+ - f_i + g_i \quad \forall i \in (1, 2, ..., n) \tag{18}$$

After replacing the value of equation (16) in Model-2 we can have:

Model-3

$$min \sum_{i=1}^{n} 2d_i^+ - f_i + g_i \tag{19}$$

As g_i is constant so it can be removed from objective equation so now objective equation becomes:

$$min \sum_{i=1}^{n} 2d_i^+ - f_i \tag{20}$$

$$S.T. - f_i + g_i + d_i^+ \geq 0 \quad \forall i \in (1, 2, ..., n) \tag{21}$$

$$d_i^+ \geq 0 \quad \forall i \in (1, 2, ..., n) \tag{22}$$

$$x \in F, x \geq 0, g_i \geq 0 \tag{23}$$

Model-3 can be directly formulated as weighted goal programming problem where positive deviation of goal i is d_i^+ and negative deviation of goal i is $-f_i + g_i + d_i^+$. Now for weighted goal programming we need to assign some weights to each deviation so w_i^+ is weight for positive deviation and w_i^- is weight for negative deviation. After rewriting the equations in weighted form:

Model-4

$$min \sum_{i=1}^{n} [(w_i^+ + w_i^-)d_i^+ - w_i^i f_i], \tag{24}$$

$$S.T. - f_i + g_i + d_i^+ \geq 0 \quad \forall i \in (1, 2, ..., n) \tag{25}$$

$$d_i^+ \geq 0 \quad \forall i \in (1, 2, ..., n) \tag{26}$$

$$x \in F, x \geq 0, g_i \geq 0 \tag{27}$$

5) Applying derived goal programming model and solving it:
For applying model 4, we first need to have some information about weights.

In equation-1:
$w_1^- = 0$, We don't have any problem if the total time of our optimal solution goes below T_T
$w_1^+ =$ some positive number

In equation-2:
$w_2^- = 0$, Reason is same as equation-1
$w_2^+ =$ some positive number

In equation-2:
$w_2^- =$ some positive number
$w_2^+ = 0$, In this case we want to maximize Y_{ij} so it will not be a problem if optimal solution goes above M_{min}

So by using these values of weights we can right our model as given below:

$$min \quad w_1^+ d_1 + w_2^+ d2 + w_3^- (d_3 - \sum_{i=1}^{n} \sum_{j=1 i \neq j}^{n} Y_{ij}) \tag{28}$$

$$S.T. - f_1 + T_T + d_1^+ \geq 0 \tag{29}$$

$$S.T. - f_2 + C_T + d_2^+ \geq 0 \tag{30}$$

$$S.T. - f_3 + M_{min} + d_3^+ \geq 0 \tag{31}$$

$$d_i^+ \geq 0 \quad \forall i \in (1, 2, 3) \tag{32}$$

$$Y_{ij} \in 0, 1 \quad \forall i \in (1, 2, .., n) \quad \forall j \in (1, 2, ..., n) \tag{33}$$

Our final model includes equations (26)-(31) With remaining constraints equation (4)-(6) as it is. This is final goal programming model of our problem that we will be using. We now have a mixed integer linear programming problem as $d_i \forall i \in (1, 2, 3) \in R^+$ and other variables have binary integer constraints.

A mixed integer linear programming problem can be solved by using simplex algorithm and branch bound method. For solving the problem with BB method we use relaxation techniques and simplex algorithm for getting initial solutions and then use this solution as starting point for BB. But to solve it we need the value of weights, that is priorities of the objectives.

6) Finding weights for solving weighted goal programming model:
Below are 2 methods for finding weights:
a. Using dual equations.
b. Using game theory model.

a) Solution using dual equations:
The dual of a general linear programming problem can be written as given below. Let the primal problem be:
Primal

$$Maximize \sum_{j=1}^{n} c_j x_j \tag{34}$$

Subject to:

$$\sum_{j=1}^{n} a_{ij} x_j \leq b_i \quad \forall i \in (1, 2, ..., m) \tag{35}$$

$$x_j \geq 0 \quad \forall j \in (1, 2, ..., n) \tag{36}$$

Associated with this primal problem there is a corresponding dual problem given by:

Dual

$$Minimize \sum_{i=1}^{m} b_i x_i \tag{37}$$

Subject to:

$$\sum_{i=1}^{m} a_{ij} y_i \geq c_j \quad \forall j \in (1, 2, ..., n) \tag{38}$$

$$y_i \geq 0 \quad \forall i \in (1, 2, ..., m) \tag{39}$$

So the dual of our weighted goal programming model can be written as:

$$min \left\{ T_T y_1 + C_T y_2 + M_{min} y_3 + \sum_{i=4}^{3n+3} y_i \right\} \tag{40}$$

Subject to:

$$w_1^+ - y_1 \leq 0 \tag{41}$$

$$w_2^+ - y_2 \leq 0 \tag{42}$$

$$w_3^+ - y_3 \leq 0 \tag{43}$$

$$\sum_{i=1}^{n} \sum_{j=1}^{n} T_{ij} y_1 + C_{ij} y_2 + y_3 + y_{(3+i)} + y_{(3+n+j)} + y_{(3+2n+i)-y_{(3+2n+j)}} - w_3^- \leq 0 \tag{44}$$

Above linear programming problem can be solved for all y_is and weights. After solving this we will get some base values of w_is which will allow us to solve the derived goal programming model.

b) Solution using game theory model:

The above method gives us some initial values of weights to start with but it is still difficult to vary the weights without knowing decision maker's preferences. This game theory method allows us to solve multi-objective linear programming problems without knowing the preferences of decision makers. Game theory is useful for making decisions in cases where two or more decisions makers have conflicting interest. This theory is divided into some categories. With number of people, it can be categorize as two person and n-person game. With gains, it can be categorize as zero-sum and non-zero-sum game. Among strategies, only one strategy may be used that is named as pure strategy or all strategies may be used with some proposition that is names as mixed strategies. In this two-person zero-sum game with mixed strategies will be used. In this type of game, there are two players. Two players have their own strategies and use all of them with different ratios. Player I has m strategies and the ratio of using them are x1 , x2 , ...,xm ; player II has n strategies and the ratio of using them y1 , y2 , ..., yn . Player Is gain is equal to player IIs loss. Table I shows pay-off matrix of the game that is formed with the player Is gain, player IIs loss as. Player I uses own strategies to maximize own gain, player II uses own strategies to minimize own loss.

Player-1	Player-2 Strategies					
Strategies	y1	y2	....	....	....	yn
x1	a11	a12				.
x2	a21	a22				.
.	....	....		aij		
xm	am1	am2				amn

Two-person zero-sum game can be used to determine weights of objectives to solve multi-objective linear programming problems with weighted sum method. In this approach Player I strategies are objectives; Player IIs strategies are the best solutions of objectives under optimizing them individually under problem constraints. If problem has K objectives, two players have K strategy. Payoff table is formed as game theory. Because of values of objectives, pay-off matrix may include both small and great values. For example maximizing product reliability-first objective is varied from one and zero, maximizing profit-second objective is varied from millions. So in game, row 2 dominates row 1. So pay-off matrix must be normalized. By normalized pay-off matrix, normalized weights are determined. Then transformation on normalized weights is made to obtain weights of objectives.

General form of our model before applying weighted goal programming model

$$max f_k(x) \quad \forall k \in (1, 2, ..., n) \tag{45}$$

Subject to:

$$S = \{Ax = b, x \geq 0, b \epsilon R^n\} \tag{46}$$

B = The set of weighting vectors

$$B = \lambda | \lambda \epsilon R^k, \lambda_i \geq 0, \sum_{i=1}^{k} \lambda_i = 1 \tag{47}$$

So the above given linear programming problem can be as

$$max \left\{ f_{k+1} = \sum_{i=1}^{k} \lambda_i f_i(x) | x \epsilon S \right\} \tag{48}$$

Now we will compute λ_i weights using game matrix and weighted sum of the function. We will obtain x_i^* extreme points and corresponding function values maximizing all $f_i(x_i)$ on S. Game matrix G is formed with these values. The game matrix G can be written as:

G =

	X1	x2	....	xj	....	xk
f1	f11	f12	....	fij		fk
f2	f21	f22	....	f2j		f2k
	⋮					
fk	fk1	fk2	....	fkj		fkk

Normalized matrix G_n can be made from Z_{ij} where

$$Z_{ij} = \frac{f_{ij}}{f_{ii}} \quad \forall j \in (1, 2, ..., k) \tag{49}$$

G_n entries represent the relative expectancy of each objective with respect to the maximum value of each objective. Weights λ_i can be calculated on normalized matrix by LP. After solving the LP formed from normalized matrix we will get the values of λ_i. But these values are not corresponding to our G matrix so for getting optimal weights for G matrix following calculations are required.

$$m_i = \frac{\lambda_i}{f_{ii}}, M = \sum_{i=1}^{k} m_i, \lambda_i^* = \frac{m_i}{M} \quad \forall i \in (1, 2, ..., k) \tag{50}$$

Here λ_i^* is the optimal weight vector. By using these weights we write composite function as

$$f_{k+1} = \sum_{i=1}^{k} \lambda_i^* f_i \tag{51}$$

This function is solved by linear programming on S and efficient point x_i^* can be obtained. The optimal solution vector corresponding to x_{k+1}^* is

$$F^* = \left[f_1(x_{k+1}^*), f_1(x_{k+1}^*) ..... f_k(x_{k+1}^*) \right] \tag{52}$$

After this we create a decision matrix D such that elements of D can be calculated as follows:

$$D = 1 - z_{ij, i \neq j} 0, i = j \tag{53}$$

from D we calculate $max(i)[max(j) d_{ij}] = d_{ij}^*$. The extreme point corresponding to d_{ij}^* is replaced by x_{k+1}^* which is extreme points of f_{k+1} and the new game matrix is found and again the same procedure is followed until we start getting same results as previous ones.

B. Data Collection and Graph Formation

For the next part of the project, we have started working on data collection and graph formation. Graph formation is an important part of the project on which all the algorithms will be tested. It consists of two parts.

1. nodes (Tourist cities in given radius/area)
2. Edges (All possible transportation modes between the cities)

We have developed some sample scrapers by using Javascript and different APIs. Following sections will describe complete APIs and scraper details.

1) Collecting data of nodes:

We are using Google Places API for collecting information about places. This API takes latitude,longitude information to get the starting point and radius up to 50,000 meters. Using the location and radius it provides the data of places in the given radius from the given point. We send a HTTP request, and get the response in JSON format. This file contains names of the places,formal address, latitude,longitude and some other information from which we extract the address first and then the city from the address. Using the above extracted data we compile a list of all cities which can be used as nodes for our final graph. Following is the pseudo code of the same.

Algorithm 1: Pseudo code for node scraper

```
Include all libraries;
url1 = 'https://maps.googleapis.com/maps/api/place/nearbysearch/fileType(xml/json)?/';
location = 'latitude,longitude';
radius = 'r';
type = 'Points of Attraction';
key = 'provided by google' URL = url1 + location + radius + type + key;
cityList = [];
request(URL);
if !error then
    JsonObject = response of request;
    i=0;
    while i≤ JsonObject.length()do
        addr = JsonObject[i].address;
        city = addr.substring(startPoint,endPoint);
        cityList.push(city);;
    end
else
    console.log(error);
end
```

2) Collecting data of edges:

Different transportation modes will work as edges between nodes. To collect this data we need some source which can provide this kind of information by taking the places as inputs. Currently we are using "Rome2Rio.com" to get the data of edges, Node.js, request(to make HTTP request),Cheers.io(to load HTML page) and body-parser(to parse the html page). To make the request we need to get the URL first. By observing the URL of Rome2Rio we can easily find out how the URL will change with the places. By using the list of nodes we create URLs then send the request and load html page using cheers.io. By observing the HTML page we get names of classes for mode, price and time which provides us the required data. Finally using body-parser we get the data and put them in JSON object to send back on the client side. Below is the pseudo code of the scraper.

Algorithm 2: Pseudo code for edge scraper

```
Include all libraries;
url1 = 'http://www.rome2rio.com/s';
from = '/place1';
to = '/place2';
URL = url1 + from + to;
request(URL);
if !error then
    load(html);
    scrapeClasses = ["modeClass","timeClass","priceClass"];
    len = scrapeClasses.length();;
    while len≥ 0do
        data = filter(scrapeClasses[len]);
        put it in multidimensional vectors and create a json object;
    end
else
    console.log(error);
end
```

VI. Conclusion

We have successfully formed a algorithm using goal programming. For formation of the algorithm we modelled single entry single exit system and general system. We then solved the derived model using two different methods i.e. dual and weighted sum game theory approach. Further we have started working on data collection. We developed some sample scrapers and with them we are able to create the graph for our problem for testing the algorithms. For future work, we will implement, test and compare the given solution methods.

VII. References

1. Fariborz Jolai*, Mehdi Aghdaghi, A goal programming model for single vehicle routing problem with multiple routes,Journal of Industrial and Systems Engineering,Vol. 2, No. 2, pp 154-163,Summer 2008

2. J.K. Sharma and M.M. Sharma, On mixed integer goal programming problems, Indian J. pure appl. Math, 11(3): 314-320, March 1980

3. Chin-Nung Liao, A Zero-one Goal Programming Model for Marketing Project Selection, Journal of China Institute of Technology Vol.40-2009.06

4. Shinto K.G1, C.M. Sushama2 An algorithm for solving integer linear programming problems, IJRET: International Journal of Research in Engineering and Technology eISSN: 2319-1163 — pISSN: 2321-7308

5. X.-Q. Li, B. Zhang and H. Li,"Computing efficient solutions to fuzzy multiple objective linear programming problems", Fuzzy Sets and System, 2006, Vol. 157, pp. 1328-1332.

6. Gulcin Dinc Yalcin, Nihal Erginel, Determining Weights in Multi-Objective Linear Programming under Fuzziness, WCE 2011, July 6 - 8, 2011, London, U.K.

7. M. O. Wankhade, H. S. Lunge, Effect Of The Changes In The Weights On The Solution Of The Preemptive Weighted Linear Goal Programming Problems, International Journal Of Scientific Technology Research Volume 3, Issue 3, March 2014.

8. U.C.Orumie1, D.W Ebong2, An Efficient Method of Solving Lexicographic Linear Goal Programming Problem , International Journal of Scientific and Research Publications, Volume 3, Issue 10, October 2013

9. Y. Lai and C. Hwang, Fuzzy multiple objective decision making:methods and applications. Newyork : Springer, 1994.