International Journal of Computer Science and Engineering (IJCSE)
ISSN(P): 2278-9960; ISSN(E): 2278-9979
Vol. 3, Issue 5, Sep 2014, 23-36

International Academy of Science, Engineering and Technology
Connecting Researchers; Nurturing Innovations

OPTIMIZATION OF TRAVELING SALESMAN PROBLEM FOR AN ENTERPRISE RESOURCE PLANNING SYSTEM

CHIRAG JAIN[1], RICHA SHAH[2] & ARUNA GAWADE[3]

[1,2]U. G. Student, Department of Computer, DJSCOE, Vile-Parle, Mumbai, Maharashtra, India

[3]Assistant Professor, DJSCOE, Vile-Parle, Mumbai, Maharashtra, India

ABSTRACT

Practical applications of the classical traveling salesman problem are rare because in most real world situations there are other constraints to be considered. To increase the range of useful applications, the classical structure must be modified. In this paper, we have implemented a hybrid approach that consists of an Adaptive Neuro Fuzzy Inference System and the Simulated Annealing algorithm, which determines the path based on distance and other factors.

KEYWORDS: Traveling Salesman Problem, ANFIS, ERP, Simulated Annealing, Fuzzy Inference

INTRODUCTION

The Traveling Salesman Problem (TSP) is a classical combinatorial optimization problem, which is simple to state but very difficult to solve. The problem is to find the shortest possible tour through a set of N vertices ('cities') so that each vertex is visited exactly once. This problem is known to be NP-complete, and cannot be solved exactly in polynomial time. [7]

Several solutions have been presented to solve the classical Traveling Salesman Problem. But, these solutions are not used in the real world scenario because distance is not the only constraint that determines the order in which the cities (referred to as "customers" from now on) are to be visited. Instead various other factors (for example, the market conditions of the product to be sold to the customer, the expenses of the salesman, etc.) are considered along with the distance.

In this paper, we have implemented a hybrid approach to adapt the classical TSP for an ERP System. An ERP (Enterprise Resource Planning) system covers a number of functional areas (like financial accounting, Human resources, Logistics, etc.). It provides an integrated suite of software modules that supports the basic internal business processes of a company. In an ERP system, a salesman has to visit numerous customers multiple times for various product inquiries. Thus, the path that determines the order in which the customers are to be visited can be decided using a modified solution to the classical TSP. An Adaptive Neuro-Fuzzy Inference System will be used to determine the priority of a customer using various factors.

The output from the ANFIS i.e. the priority generated will be fed along with the distance of the customer to the Simulated Annealing algorithm to determine the order in which the customers must be visited. Thus the final path will be calculated considering distance (as in the classical TSP) as well as various other factors.

RELATED WORK

Many exact and heuristic algorithms have been devised to solve the TSP. A few of these algorithms have been described as follows:

Exact Algorithms

The Brute Force algorithm is an exact algorithm, where we find all the possible solutions and then select the one with the lowest cost. Other approaches include branch and bound, progressive improvement algorithms, branch and cut, etc. The exact algorithms are designed to find the optimal solution to the TSP, that is, the tour of minimum length. The exact algorithms are typically derived from the integer linear programming (ILP) formulation of the TSP:

Min $\sum_i \sum_j d_{ij} x_{ij}$

subject to:

$$\sum_j x_{ij} = 1; \; i=1,..., N$$

$$\sum_i x_{ij} = 1; \; j=1,..., N \; (x_{ij}) \varepsilon X$$

$$x_{ij} = 0 \text{ or } 1$$

where d_{ij} is the distance between vertices i and j and the x_{ij}'s are the decision variables: x_{ij} is set to 1 when arc (i,j) is included in the tour, and 0 otherwise. $(x_{ij}) \varepsilon X$ denotes the set of sub tour-breaking constraints that restrict the feasible solutions to those consisting of a single tour.

The exact algorithms determine the best path, given a number of cities. The computation time is negligible when the number of cities is small. But, when the number of cities increases, so does the computation time – it is of the order of n!, where n is the number of cities.

Heuristic Algorithms

Finding the optimum solution to the TSP using an exact algorithm may take less time on an expensive computer. But, it is more cost effective if we find a "near-optimal" solution on a computationally less powerful computer. Hence, heuristic or approximate algorithms are often used in place of exact algorithms for solving large TSPs. TSP heuristics can be classified as tour construction procedures, tour improvement procedures, and composite procedures, which are based on both construction and improvement techniques. [1]

- **Construction Procedures**

Procedures in this class build the path by selecting one vertex at a time and inserting it into the current path one by one. For selecting the next vertex and identifying the best place to insert it, various factors are considered like proximity of current tour and the minimum tour.

- **Improvement Procedure**

The k-opt exchange heuristics - in particular, the 2-opt, 3-opt, and Lin-Kernighan heuristics are the most widely used amongst the local improvement procedures. These heuristics techniques locally modify the current solution by replacing k arcs in the tour by k new arcs so as to generate a new improved tour. Typically, the exchange heuristics are

applied iteratively until a local optimum is found, which cannot be improved further by using the exchange heuristic under consideration.

- **Composite Procedures**

Here, the tour construction procedure is a simple greedy heuristic. In the beginning, each city is considered as a fragment, and multiple fragments are built in parallel by iteratively connecting the closest fragments together until a single tour is generated. The solution is then processed by a 3-opt exchange heuristic.

Artificial Neural Networks

A number of neural networks have been used to solve the classical TSP. Some examples have been described as follows:

- **Hopfield Tank Neural Network**

The original Hopfield neural network model is a fully interconnected network of binary units with symmetric connection weights between the units. The connection weights are not learned but are defined a priori from problem data (the inter-city distances in a TSP context). Starting from some arbitrarily chosen initial configuration, either feasible or infeasible, the Hopfield network evolves by updating the activation of each unit in turn (i.e., an activated unit can be turned off, and an inactivated unit can be turned on). Through this update process, various configurations are explored until the network settles into a stable configuration. In this final state, all units are stable according to the update rule and do not change their activation status. [5]

- **Elastic Net**

The elastic net algorithm is an iterative procedure where M points, with M larger than the number of cities N, are lying on a circular ring or "rubber band" originally located at the center of the cities. The rubber band is gradually elongated until it passes sufficiently near each city to define a tour. During that process two forces apply: one for minimizing the length of the ring, and the other one for minimizing the distance between the cities and the points on the ring. These forces are gradually adjusted as the procedure evolves. [6]

The disadvantage of these methods used to solve the classical TSP, is that they consider only distance as a parameter to determine the path. In the real world, a salesman decides the order in which he should visit the customers based on a number of parameters. Distance is just one of the many factors considered.

PROPOSED SOLUTION

The various methods that were studied could not be easily adapted to consider other factors along with distance. Hence, we decided to first use an Adaptive Neuro-Fuzzy Inference System where we give other factors that are considered to determine the order in which the customers are to be visited. This system will generate a priority for each customer, which will be combined with distance, and given to Simulated Annealing algorithm to finally determine the path.

Thus, this paper proposes the following hybrid approach towards solving the Traveling Salesman Problem for an ERP System. This approach is composed of two phases – ANFIS, followed by Simulated Annealing.

Phase I (ANFIS)

Adaptive Neuro-Fuzzy inference system (ANFIS) is a kind of neural network that is based on Takagi–Sugeno fuzzy inference system. Since it integrates both neural networks and fuzzy logic principles, it has potential to capture the benefits of both in a single framework. Based on the inputs, it was easier to use an ANFIS where a set of fuzzy IF-THEN rules can be used to determine the priority of each customer.

Architecture

The ANFIS architecture consists of six layers namely, Inputs, IF-Part, Rules, Normalization, THEN-Part, and Output. The output generated by one layer is fed to next layer where each layer performs a specific task which is described below.

Layer 0: Inputs to ANFIS

The following inputs will have a numeric value that is fed into the Adaptive Neuro-Fuzzy inference system. The values can then be mapped to Low or High depending upon the value of input. Further, the range of values (Universe of discourse) within which these inputs lie varies.

- Market Conditions for the Product
- Average Discount given for Product
- Quantity of Product to be sold
- Expenses for a Customer
- Probability of successful deal
- Cost Price of Product

Layer 1: Evaluating Membership (IF-Part)

For all inputs Gaussian membership function is used to calculate the membership value.

The formula for Gaussian membership function is:

$$f(x;\sigma,c) = e^{\frac{-(x-c)^2}{2\sigma^2}}$$

The graph of membership value against the input can be as follows

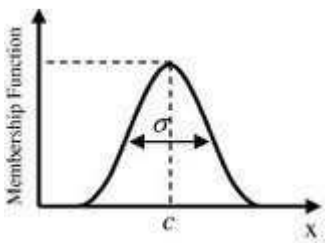

Figure 1

The values for c and σ are as follows:

Table 1

	Low		High	
	c	**σ**	**c**	**σ**
Market Conditions	-100	60	100	60
Avg. Discount	0	20	100	25
Product Quantity	0	25	100	20
Expenses	0	25	100	25
Probability of Success	0	25	100	25
Product Price	0	35	100	30

The output calculated can be represented as

$$O_{1,i} = \mu_{A_i}(x)$$

Here x is the input value and $\mu_{A_i}(x)$ is the corresponding membership function.

Layer 2: Rules

Every node in this layer is fixed. This is where the t-norm is used to 'AND' the membership grades - for example the product:

$$O_{2,i} = w_i = \mu_{A_i}(x)\mu_{B_i}(y), \quad i = 1,2$$

Layer 3: Normalization

It contains fixed nodes which calculate the ratio of the firing strengths of the rules as follows:

$$O_{3,i} = \overline{w_i} = \frac{w_i}{w_1 + w_2 + \ldots + w_i}$$

Layer 4: Consequent Parameter Calculation (THEN-PART)

The nodes in this layer are adaptive and perform the consequent of the rules:

$$O_{4,i} = \overline{w_i} f_i$$

The graph of membership value against the input can be as follows

The values for c and σ are as follows:

where f_i is a function of input values which can be expressed as:

$f_i = p_1x_1 + p_2x_2 + p_2x_2 + p_3x_3 + p_4x_4 + p_5x_5 + p_6x_6 + r$ where x_1 denotes market conditions, x_2 denotes average discount, x_3 denotes product quantity, x_4 denotes average expenses, x_5 denotes probability of successful deal, x_6 denotes product price, p_1 to p_6 and r denote the consequent parameters.

Since we consider 64 rules, there are 64 different values for p_1 to p_6 and r.

Layer 5: Output

There is a single node here that computes the overall output as follows:

$$O_{5,i} = \sum_i \overline{w_i} f_i = \frac{\sum_i w_i f_i}{\sum_i w_i}$$

The ANFIS architecture is as shown below. In our approach, there will be six inputs as described previously, along with 64 rules. The input vector is fed through the network, layer by layer.

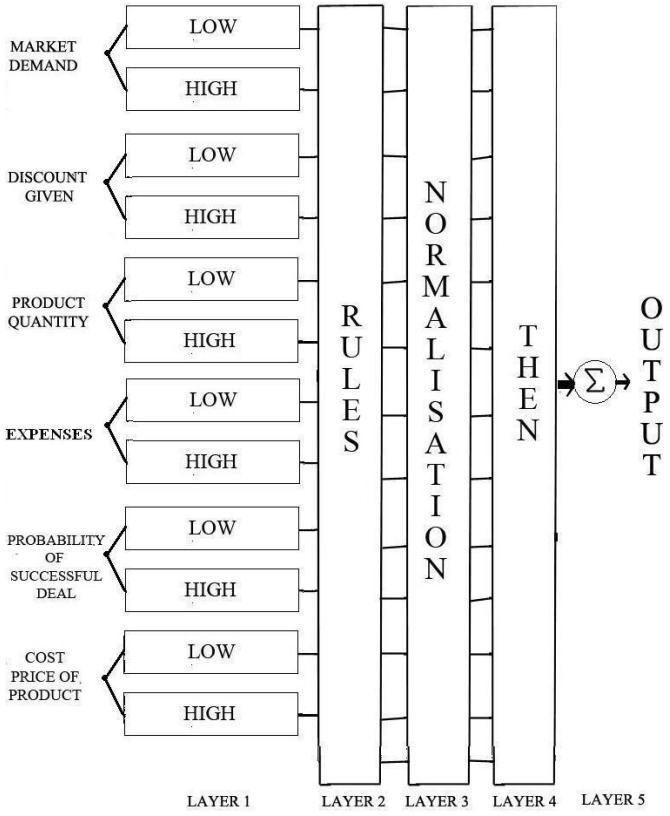

Figure 2: ANFIS Architecture

The Gradient descent algorithm has been used to update the values of p1, p2, p3, p4, p5, p6 and r which are used in the function at Layer 4. To find a local minimum of a function using gradient descent, one takes steps proportional to the negative of the gradient (or of the approximate gradient) of the function at the current point.

Gradient descent is based on the observation that if the multivariable function $F(\mathbf{x})$ is defined and differentiable in a neighborhood of a point a, then $F(\mathbf{x})$ decreases fastest if one goes from **a** in the direction of the negative gradient of

$F(\mathbf{x})$ at a, $-\nabla F(\mathbf{a})$. It follows that, if

for γ small enough, then $F(\mathbf{a}) \geq F(\mathbf{b})$. With this observation in mind, one starts with a guess $\mathbf{x}_0$ for a local minimum of F, and considers the sequence $\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, \ldots$ such that

$$\mathbf{x}_{n+1} = \mathbf{x}_n - \gamma_n \nabla F(\mathbf{x}_n), \ n \geq 0.$$

We have

$$F(\mathbf{x}_0) \geq F(\mathbf{x}_1) \geq F(\mathbf{x}_2) \geq \cdots,$$

so hopefully the sequence (x_n) converges to the desired local minimum.

Output

The ANFIS will give a numerical output which will represent the priority of the corresponding customer.

Phase II (Simulated Annealing)

Simulated annealing (SA) is a generic probabilistic meta- heuristic for the global optimization problem of locating a good approximation to the global optimum of a given function in a large search space. The key idea in simulated annealing algorithm is to select an appropriate temperature schedule which needs to specify the initial, relatively large value of T and then decrease the value of T. Starting with relatively large values of T, the probability of acceptance is relatively large, which enables the search to proceed in almost all random directions. Gradually decreasing the value of T as the search proceeds gradually decreases the probability of acceptance, which emphasizes on climbing upwards.

Simulated Annealing has been used to solve the classical TSP. It has been used in this hybrid approach because it can be easily modified to consider the priority of each customer, combined with distance in order to determine the path.

Inputs to Simulated Annealing

The latitude and longitude of the customers, along with their corresponding priorities will be given to the Simulated Annealing algorithm.

Acceptance

The probability of transiting from current state X_n to a current trial solution X'_n is specified by an acceptance probability function P (e, e', T) which depends on the energies e = E (X_n) and e' = E (X'_n) of the two states and a global varying parameter called the Temperature. Essential points for designing P are:

- Must be nonzero when e'> e, meaning the system might move to the new state X'_n even if it is worse than the current one. This feature prevents the method from being stuck in local minimum.
- When T goes to zero, P (e, e', T) must tend to be zero when e'> e and to a positive value if e'< e.

The flowchart for the simulated annealing algorithm is as shown below [4]

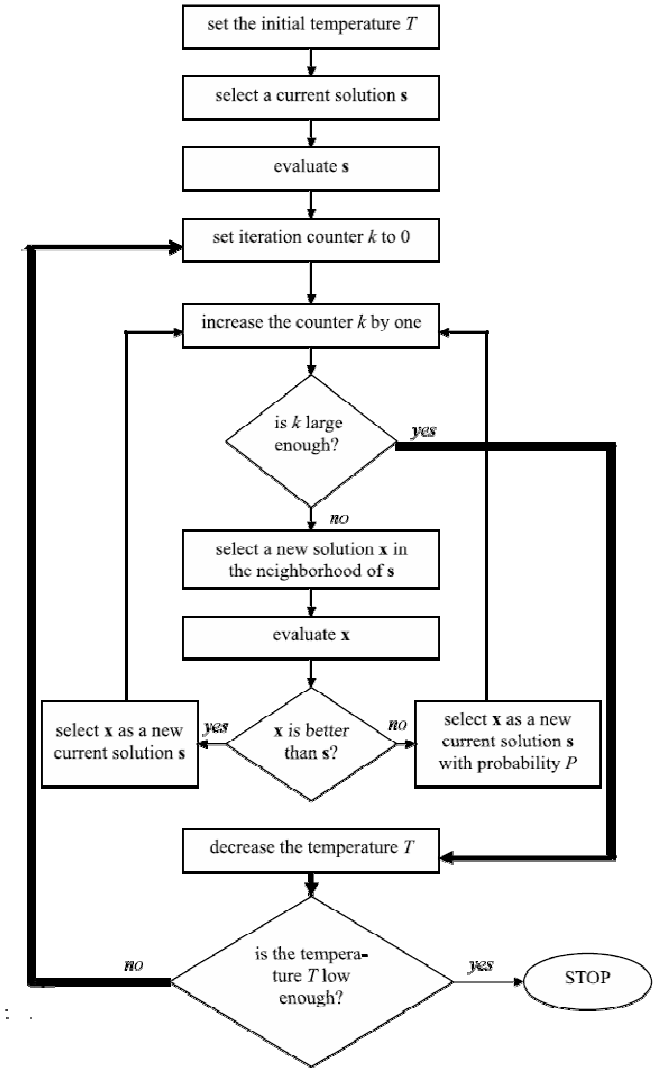

Figure 3: Simulated Annealing Flowchart

Algorithm

X_n is the solution after n iterations.

X'_n is the current trial solution.

Temperature (T) is the parameter that measures the tendency to accept the current solution as next trial solution.

The 'move selection rule' is then based on selecting which neighbor will be the next trial solution.

- Create the initial list of cities by shuffling the input list (i.e.: make the order of visit random).
- Select X'_n neighbor of X_n
- The cost value is the distance travelled by the salesman for the whole tour.
- If neighbor is 'better' in cost i.e. $f(X'_n)$ is less than $f(X_n)$, then accept with probability p_n, $X_{n+1} = X'_n$.
- If neighbor is 'worse' in cost i.e. $f(X'_n)$ is greater than $f(X_n)$, then accept with probability p_n, $X_{n+1} = X'_n$ or reject with probability $(1 - p_n)$, $X_{n+1} = X'_n$. We could have

$$p_n = e^{\frac{-(f(X'_n) - f(X_n))}{T}}$$

- We update the temperature during every iteration by slowly cooling down.
- We let Temperature (T) go to zero over time.

The probability of accepting a worse state is given by the equation: [3]

P = exp(-c/t) > r

Where

c = the change in the evaluation function

t = the current temperature

r = a random number between 0 and 1

The evaluation function is a function of the distance between customers and their priorities. The probability of accepting a worse move is a function of both the temperature of the system and of the change in the cost function. As the temperature of the system decreases the probability of accepting a worse move is decreased. Thus, if the temperature is zero then only better moves will be accepted.

Output

The Simulated Annealing algorithm will give the order in which the customers are to be visited as the output.

RESULTS

TSP Optimizer is a software programmed in C#.NET framework, which retrieves the details of inquiries from an online database in MySQL using PHP pages on server. When the manager logs into the software, he/she can view all the inquiries currently in the database. The manager can also view details of selected inquiries. After selecting inquiries, the manager can proceed to the next stage where processing by ANFIS will take place. After the ANFIS completes processing, the manager can view the intermediate results, which will show the priorities of each inquiry in a decreasing order.

After the ANFIS stage is complete, the manager moves onto the Simulated Annealing stage, where a number of rounds will be performed, in order to find the near optimal path with least cost. After the Simulated Annealing completes processing, the manager can view the path on a Google Map, and can also view the detailed results in another window.

The manager can also store the processed results at various stages in an Excel sheet. Snapshots of the GUI are as shown below:

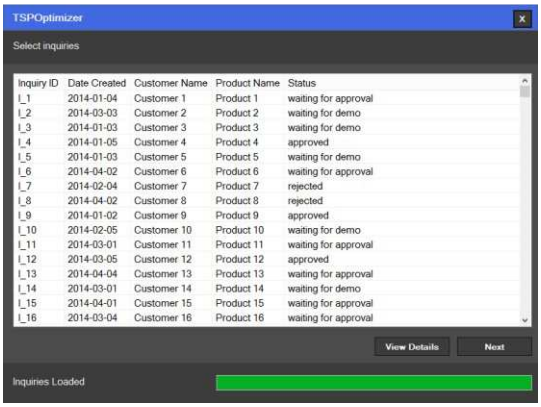

Figure 4

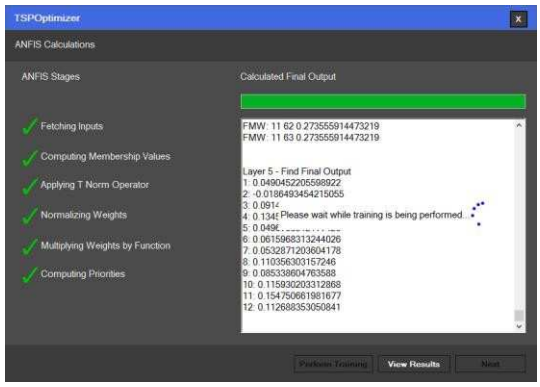

Figure 5

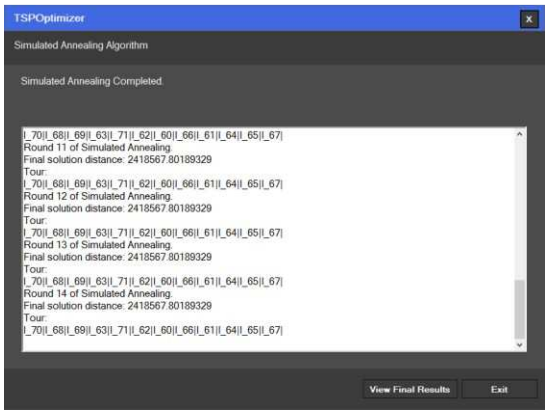

Figure 6

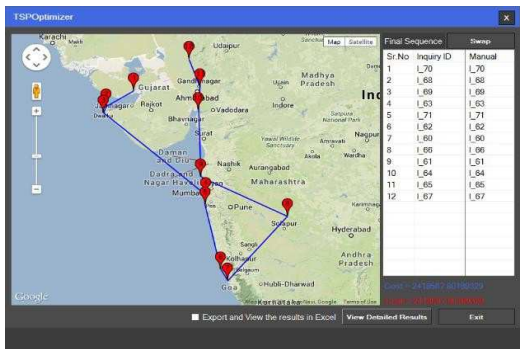

Figure 7

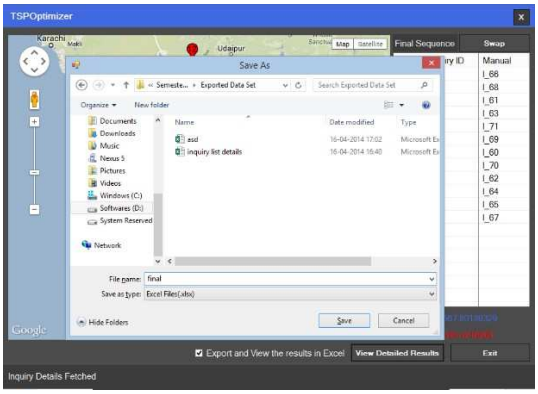

Figure 8

The software was run on both 32-bit and 64-bit computers in order to compare the run time for the ANFIS and Simulated Annealing stages individually.

The results are as follows:

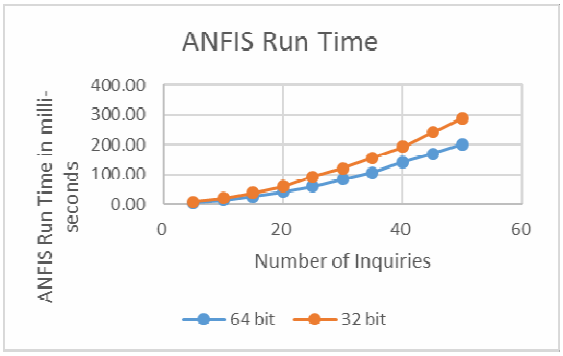

Figure 9

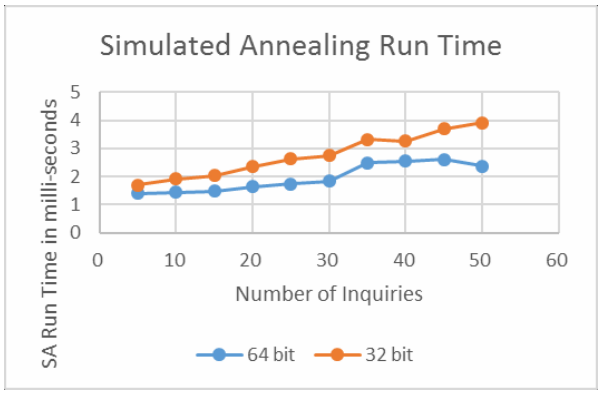

Figure 10

As inferred from the graphs above, ANFIS takes much greater time as compared to Simulated Annealing processing. Also, the software was run five times simultaneously for specific values of k, T_0, and cooling rate and the cost values were recorded as shown.

The lowest average cost was recorded when the cooling rate was 0.001. As the values of k, T_0 and cooling rate increase, the average cost reduces, but, the run time of the software increases.

Table 2

k	T0	Cooling Rate	Average Cost Over 5 Rounds
15	10000	0.001	3022438.2
15	10000	0.003	3060782.4
15	10000	0.006	3522728.4
20	10000	0.001	2949682.8
20	10000	0.003	3256344.4
20	10000	0.006	3723838.8
25	10000	0.001	3199557.2
25	10000	0.003	3286320
25	10000	0.006	3321001.4
15	15000	0.001	2897974.4
15	15000	0.003	3286421.6
15	15000	0.006	3324775
20	15000	0.001	2932695.6
20	15000	0.003	3077394
20	15000	0.006	3137019.8
25	15000	0.001	2934944
25	15000	0.003	3195524.6
25	15000	0.006	3268483.2

CONCLUSIONS

TSP Optimizer is a software designed for the use of a manager or any other high-level member of an organization that uses an ERP System. The database used in the system has been designed while keeping an ERP model in mind. The software gives the manager a number of options, from generating a priority for selected inquiries to generating the path for the same. The software also allows the manager to export the data generated at various stages of processing.

TSP Optmizer aims to bridge the gap between the classical Traveling Salesman Problem and the real world scenario, by considering factors other than distance to generate the final path. Adaptive Neuro-Fuzzy Inference System and Simulated Annealing have been used in order to get the near optimal solution by taking customer and product related factors also into consideration for calculations and not just distance between customers.

It can be easily integrated into any system that uses the ERP model. The system's main use is eliminating the complex calculations involved in deciding the order in which the inquiries are to be visited. It helps in saving time by generating not only the path, but also priorities for multiple inquiries.

FUTURE SCOPE

The software has a huge future scope because ERP is used in a lot of industries today. Integration with an actual ERP system will only involve changes in the database model that we have considered.

Based on an organization's needs, the factors that have been considered as an input to the ANFIS can be varied. The number of factors can also be increased or decreased.

Our algorithm has been designed to generate the near optimal path with the least cost. In order to increase the efficiency of the system, the processing can be decentralized. The code can also be optimized for target computers (32-bit or 64-bit) so that the intense calculations can fully utilize the computer resources.

Google Maps API can be fully used in order to use the actual road distance while calculating the cost for a path. Further, the map can be optimized in order to show the path with driving directions. After the path has been finalized, the data generated can be used to directly book tickets for the salesman via a partner traveling agency.

The data generated at various stages can also be stored in a warehouse for future analysis and mining.

REFERENCES

1. Jean-Yves Potvin, "The Traveling Salesman Problem: A Neural Network Perspective", ORSA Journal on Computing 5, 328-348, 1993.
2. Jyh-Shing Roger Jang, Chuen-Tsai Sun and Eiji Mizutani, "Neuro- Fuzzy and Soft Computing: A Computational Approach to Learning and Machine Intelligence"
3. S. N. Sivanandam, S. N. Deepa, "Principles of Soft Computing"
4. Zbigniew Michalewicz, Martin Schmidt, Matthew Michalewicz, Constantin Chiriac, "Adaptive Business Intelligence"
5. Jacek Ma´ndziuk, "Solving the Travelling Salesman Problem with a Hopfield - type neural network", Institute of Mathematics, Warsaw University of Technology
6. Donald Davendra, "Traveling salesman problem, theory and applications", In Tech December 2010
7. http://en.wikipedia.org/wiki/Traveling_salesman_problem