Atlantis Highlights in Engineering (AHE), volume 1
International Conference on Science and Technology (ICST 2018)

Comparison of the Traveling Salesman Problem Analysis Using Neural Network Method

Aeri Rachmad[1], Eka Mala Sari Rochman[2], Dwi Kuswanto[3], Iwan Santosa[4]
Faculty of Engineering
University of Tronojoyo Madura
Bangkalan, Indonesia
[1]aery_r@yahoo.com, [2]ekamala.sari@yahoo.com,
[3]dwikuswanto@yahoo.com, [4]iwan@trunojoyo.ac.id

Rinci Kembang Hapsari[5], Tutuk Indriyani
Faculty of Information Technology
Adhi Tama Institute of Technology Surabaya
Surabaya, Indonesia
[5]rincikembang@itats.ac.id

Endah Purwanti[6]
Faculty Sains Technology
Airlangga University
Surabaya Indonesia
[6]endah-p-1@fst.unair.ac.id

***Abstract*—Traveling sales problem (TSP) is one of the classical optimization problems and NP-complete. The Challenge in implementing TSP is how to determine the shortest distance from a traveling route of N cities where each N city visited precisely once at time. In this paper, we attempt to analyze the completion of TSP using an artificial neural network approach. In network, weights are determined to represent problem boundaries and optimize the completion function. The approach of artificial neural networks used is a network with Hopfield algorithm and Simulated Annealing algorithm. The solution with the Hopfield algorithm has the complexity of the $4n^4 + 16n^3$ algorithm, while the complexity of the Simulated Annealing algorithm is 10n2. Judging from the use of memory in the implementation, the application of annealing algorithm is better than the Hopfield algorithm. but both are algorithms that "no works in place".**

Keywords—traveling salesman problem; optimization; hopfield; neural networks

I. INTRODUCTION

Traveling Salesman Problem (TSP) is a complete NP problem [1] and NP-hard problem [2] which is up to now only a few individual cases found could be executed polynomial with a reducible Turing machine. TSP is a typical combinatorial optimization problem [3],[4],[5], which is simple to state but very difficult to solve. The problem in TSP is the difficulty in determining the journey path that will be taken. The very concerned determination of the journey path is the total distance of the trip, which is to find the shortest possible flow of travel through a set of N cities, each of which is visited exactly once, while there is an N!/2N possible flow of travel. The greater the value of N, the bigger the likelihood of the trip path, thus it takes long time to design the desired trip plan. It happens because traveling salesmen have to look for possible ways to be traversed and decide the shortest path among all possible paths.

There have been many exact and heuristics algorithms designed a good solution. However, they vary regarding to complexity and efficiency in resolving TSPs at various levels of complexity and size (small, medium and large). Previous research used linear programming of Dantzig et al.[6], effective formulations of Held and Karp [7], but their ability was limited to small problems (less than 40 cities). Later, the artificial intelligence approach has been proven to have the ability to solve more complex issues; one of those approaches is organized neural networks [8], [9], and then expanded as metaheuristic. A metaheuristic can optimize complex problems [10]. The way metaheuristic is done in solving problems is to search through many candidate solutions with little or no assumptions about the problem being solved and without any guarantee to find the optimal solution.

Recently there have been many optimization problems that can be solved using artificial neural networks [11]. An artificial neural networks approach, can solve TSP problems quickly [12]. Therefore, in this paper, the completion of TSP will show and compare two artificial neural network algorithms, namely the Hopfield algorithm and the Simulated Annealing algorithm.

II. METHOD

A. Traveling Salesman Problem

The definition of the traveling salesman problem is given n city and d_{ij} which is the distance between c_i cities and c_j cities. A salesman wants to make a closed track by visiting each city only once and the journey begins and ends in the same city[13], [14], [15], [16]. TSP has a problem limitation in two conditions, namely weak limitation and strong limitation [17].

Weak problem limitation. In this condition, the completion obtained will not be wrong even though the limitation of this problem is not fulfilled. Still, the limitation of the problem must be fulfilled whenever possible. The example is a path with minimum mileage.

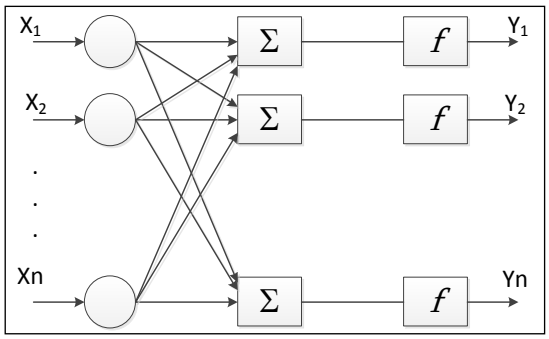

Fig. 1. Model of Single Layer Neural Network

Strong problem limitation. The limitation of problems must be fulfilled, otherwise the completion obtained will be wrong. The example is that every city listed in the itinerary must be visited once, not twice or more at the same time, and the journey begins and ends in the same city.

B. Artificial Neural Network

The artificial neural network is an information processing system that has specific characteristics similar to those in biological neural networks [18]. Artificial neural networks are designed as interconnection systems using elements of process called weights, each of which has many inputs and produces output.

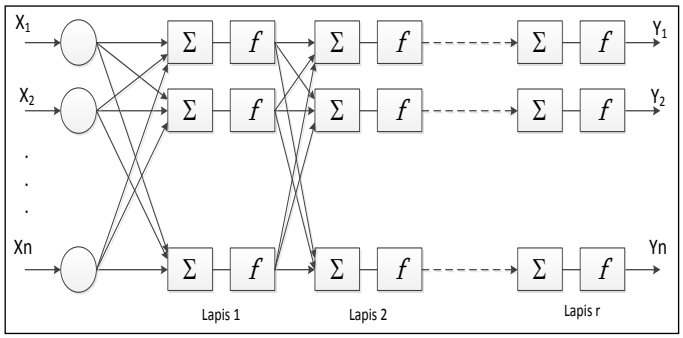

Fig. 2. Model of Multiple Layer Neural Networks

Seen from the number of layers, artificial neural networks are divided into two types, namely single layer neural network and multilayer neural network. The simplest network consists of a group of neurons arranged in one layer. The nodes on the left side of Figure 1 function only to distribute inputs and not to do calculations. Each of the X input elements is connected to each artificial neuron through a different weight.

Multiple layer neural networks have much better capabilities than single layer neural networks do. Simply put, multiple layer artificial neural networks can be made by arranging a set of single layer as shown in Fig 2.

C. Hopfield Neural Network

One of the artificial neural network architectures is the Hopfield Network. The Hopfield network consists of a number of nodes that are connected to each other[19],[20]. The connections between units have a w_{ij} weight. The value of the weight is fixed and symmetrical, meaning that the weight between neutron i to neutron j is defined as w_{ij}, equal to the weight between the neutrons j and neutrons i defined as w_{ji}, The mathematic function is as follows:

$$\omega_{\iota\varphi} = \omega_{\varphi\iota} \quad (1)$$

The steps to solve the traveling salesman problem using the Hopfield algorithm network are as follows

Step 0: Initialize the activities of all units.

Initialize Δt with a small value, $\Delta t = 10^{-5}$

Step 1: If the STOP condition is wrong, do steps 2 - 6

Step 2: Do step 3 - 5 times n2 times (n number of cities)

Step 3: Select a unit randomly

Step 4: Changes activation to select units:

$$u_{x,i}(new) = u_{x,i}(old) + \Delta t \left[-u_{x,i}(old) - A\sum_{j\neq i} v_{x,j} - B\sum_{y\neq x} v_{y,i} + C\{N - \sum_x \sum_j v_{x,j}\} - D\sum_{y\neq x} d_{x,y}(v_{y,i+1} + v_{y,i-1}\right) \quad (2)$$

Step 5 : Use the output function:

$$v_{x,i} = 0.5\left[1 + tanh(\alpha u_{x,i})\right] \quad (3)$$

Step 6 : Check stop conditions

D. Simulated Annealing

Simulated annealing (SA) can be used for the search process in optimization cases in discrete and non-existent forms. SA is widely used because of its effectiveness in finding optimal solutions to combinatory optimization problems [21]. The most important part of the SA is an annealing schedule / cooling schedule that determines the rate of decrease in temperature from the level of low temperature setting [22]. What needs to be considered in the cooling schedule process is that if the temperature drops too slowly, the CPU time will be wasted.

The experimental results are shown in Fig 5 and Fig 6. Annealing schedule regulates the decrease in temperature T as a time or iteration function. How to determine annealing schedule is to reduce the temperature T at each iteration. Where is the SA Algorithm for TSP compliance as follows

1. Step 1: Determine the starting point / solution x in the form of vector and initial temperature T, where the iteration starts from k to 1.
2. Step 2: Evaluate objective functions, E = f (x)
3. Step 3: Selection of Δx with probability determined by function g (Δx, T) Determine the point / solution of the $x_{new} = x + \Delta x$
4. Step 4: Calculate new values from objective functions: $E_{New} = f(x_{new})$

5. Step 5: Determine x as x_{new} and E become E_{New} with a probability determined by the function h (Δx, T), where $\Delta E = E_{New} - E$
6. Step 6: Reduce temperature T (determined T = ηT, where η is a constant between 0 and 1)
7. Step 7: k = k + 1, if k has the maximum iteration will stop. If you haven't reached the maximum state, repeat step 3.

III. RESULT AND DISCUSSIONS

In completing the TSP with the Hopfield method and Simulated annealing, the first thing to do is to determine a network by locating the problem into a network form, which is a quadratic array. Energy functions for TSP must meet the following requirements:

1. Every city can only be visited once on a trip
2. Each city has a spesific order of visits on the track. Two or more different cities may not have the same order.
3. All cities must be visited
4. (If possible) the trajectory is the shortest path.

Determining the efficiency of the Hopfield algorithm and Simulated annealing in solving traveling salesman problems is based on the number of main operations at the average case and the number of the worst case.

A. The Hopfield algorithm

1. Average Case

The amount of work done for input size n, which is concluded A (n), is as follows:

a. Hyperbolic tangent operation

In Hopfield algorithm, in step 5 there is a hyperbolic tangent function, where

$$tanh(\alpha u_i) = \frac{e^{\alpha u_i} - e^{-\alpha u_i}}{e^{\alpha u_i} + e^{-\alpha u_i}} \quad (4)$$

Thus the highest type of operation in this algorithm is exponential operation. Exponential basic operations are as follows:

$$A(n) = \sum_{I \in D_n} p(I)t(I) \quad (5)$$

$$= 1 * (4n^4 + 16n^3)$$
$$= 4n^4 + 16n^3$$

b. Multiplication Operations

In the multiplication operation, the element calculated is the number of the basic operation for multiplication, in which t (I) is the number of basic operations of multiplication and p (I) is the probability for input I. Because the input type is only one then p (I) is 1

$$A(n) = \sum_{I \in D_n} p(I)t(I) \quad (6)$$

$$= 1 * (6n^4 + 24n^3 + 5n^2)$$

$$= 6n^4 + 24n^3 + 5n^2$$

c. Addition Operations

In the addition operation, the element calculated is the number of the basic operation for addition, where t (I) is the sum of the basic operations of addition and p (I) is the probability for input I. Because the input type is only one then p (I) is 1.

$$A(n) = \sum_{I \in D_n} p(I)t(I) \quad (7)$$

$$= 1 * (5n^4 + 12n^3 - 8n^2)$$

$$= 5n^4 + 12n^3 - 8n^2$$

2. The number of the worst case

The number of the worst case is calculated by determining the conditions causing the biggest basic case. For this problem the condition is never found because the input type is only one. Moreover, because of the highest value of three exponential basic operations that have the highest value, the number of the worst work is equal to the average of the exponential operations, namely:

$$W(n) = \max_{I \in D} t(I) \quad (8)$$

$$= 4n^4 + 16n^3$$

From this algorithm, the variable used to store the input is array w, n which states the number of array elements, p as the abscissa and q as the ordinate. The memory used to store input is f (n) = (2n + 7)* k. In addition, there are extra variables including v_{ij}, u_{ij}, i and j. The memory used to store this extra variables is f (n) = ($n^4 + n^2 + 1$) * k, where k is a constant indicating the type of variable used. Hence, the size of memory to store extra variables depends on the size of the input. The experimental results are shown in Fig 3 and Fig 4.

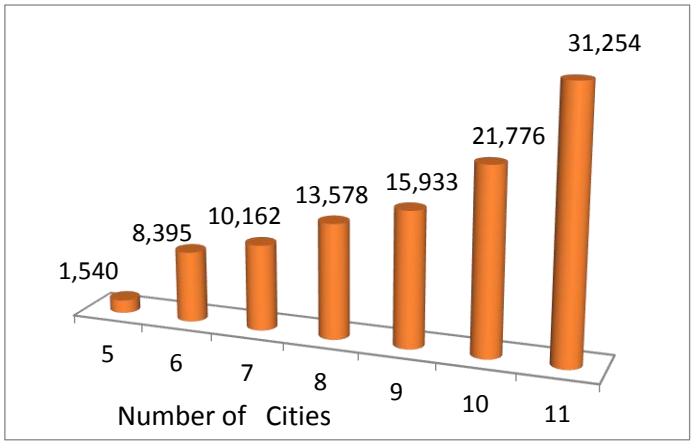

Fig. 3. Graph of test results, the relationship of the number of cities with the number of iterations using Hopfield Algorithm

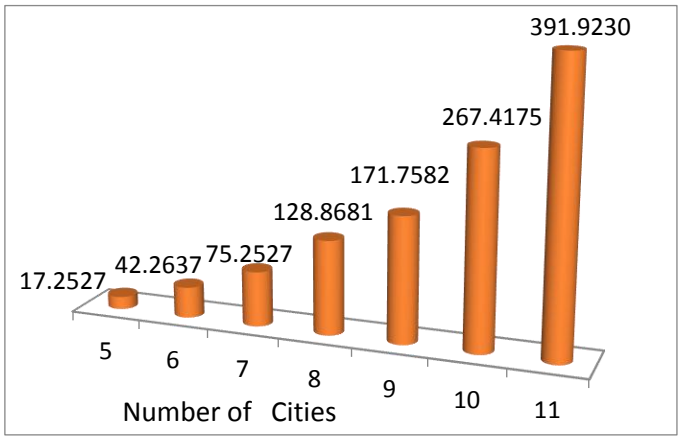

Fig. 4. Graph of test results related to the number of cities with running time (seconds) using Hopfield Algorithm

B. *Simulated Annealing*

1. *Average Case*

The amount of work done for input size n, which is concluded A (n) is as follows:

a. Exponential Operation

In this exponential operation that is calculated is the amount of exponential base operations. Because the input type is only one, the A (n) value is: $10n^2$

b. Multiplication Operations

In the multiplication operation calculated is the magnitude of the multiplication base operation, and because the input type is only one, the A (n) value is: $70n^2 + 10n$

c. Operation

In the sum operation, what is calculated is the sum of the basic operations of the sum and because the type of input is only one, the value of A (n) is: $350n^2$

2. *Number of Wrost Case*

Much of the worst work can be calculated by determining the conditions that cause the biggest basic work, for this problem the condition is never found, because the input type is only one. And because of the three exponential basic operations that have the highest value, then the worst amount of work is equal to the average of exponential operations. So that is the worst case of the Hopfield algorithm, namely:

$$W(n)=\underset{I\in D}{\text{Max}}\, t(I) \tag{9}$$

$$= 4n^4 + 16n^3$$

The worst case of the Simulated Annealing algorithm is $10n^2$

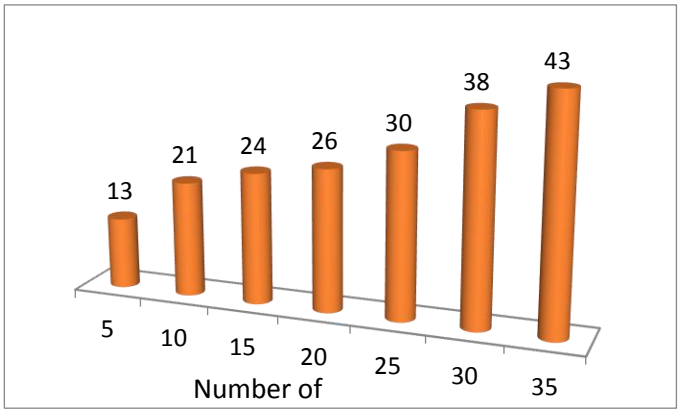

Fig. 5. Graph of the results of the trial of the SA Algorithm in relation to the number of cities with the number of iterations

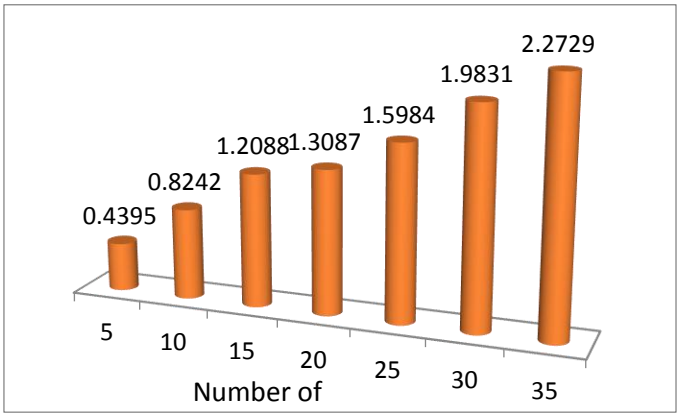

Fig. 6. Graph of the results of the trial of the SA Algorithm in relation to the number of cities with running time (seconds)

IV. CONCLUSION

From the two algorithms discussed, the Simulated Annealing algorithm is the most optimal way to complete TSP. Hopfield algorithm has a fundamental weakness, which is very dependent on the value of the coefficients. Where there is no exact set of coefficients to get the best solution. So the most widely used method is trial and error algorithm. The amount of extra space used to store in the memory of the two algorithms depends on the size of the input, so this algorithm is said to be "no works in place"

ACKNOWLEDGMENT

We thank the University of Trunojoyo Madura for providing facilities and places to complete this research. Friends of the team of Multimedia Engineering Lecturers and Networks who helped to accomplish this paper. A friend of the lecturer at the Adhi Tama Institute of Technology campus in Surabaya who gave the idea. The younger siblings of students who have given all their time to complete this research.

REFERENCES

[1] J. Y. Potvin, "The Traveling Salesman Problem: A Neural Network Perspective," 1993.

[2] G. Martino, "Solving a Traveling Salesman Problem with a Flower Structure," J. Ap-plied Math. Phys., vol. 2, pp. 718–722, 2014.

[3] X. Wang, A. Mu, and S. Zhu, "ISPO: A New Way to Solve Traveling Salesman Problem," Intell. Control Autom., vol. 4, pp. 122–125, 2013.

[4] A. Srour, Z. A. Othman, and A. R. Hamdan, "A Water Flow-Like Algorithm for the Traveling Salesman Problem," 2014.

[5] Y. I Song and N. Wei. Application of An Improved Genetic Algorithm with The Search Space Compression in TSP, Proceedings of the 2nd International Conference On Systems Engineering and Modeling (ICSEM-13), pp 0529-0532, 2013

[6] M. Jünger et al., "Solution of a Large Scale Traveling Salesman Problem," 50 Years Integer Program. 1958-2008 From Early Years to State-of-the-Art, pp. 1–804, 2010.

[7] M. Held and R. M. Karp, "A Dynamic Programming Approach To Sequences Problems," vol. 10, no. 1, 1962.

[8] T. A. S. Masutti and L. N. de Castro, "A self-organizing neural network using ideas from the immune system to solve the traveling salesman problem," Inf. Sci. (Ny)., vol. 179, no. 10, pp. 1454–1468, 2009.

[9] Torres-Jiménez, "Applications of metaheuristics in real-life problems," Progress in Artificial Intelligence, vol2, no 4, pp. 175-176, 2014.

[10] V. M. Mladenov and N. G. Maratos, "Neural Networks for Solving Constrained Optimization Problems," no. 1.

[11] S. Gupta, M. Kakkar, "Techniques For Solving Travelling Sales Man Problem, " International Journal Of Engineering Science & Advanced Technology Volume-2, Issue-5, pp.1357 – 1360, 2012.

[12] R. Pasti and L. N. de Castro, "A Neuro-Immune Network for Solving the Traveling Salesman Problem," 2006 IEEE Int. Jt. Conf. Neural Netw. Proc., pp. 3760–3766, 2006.

[13] M. Y. Kao and M. Sanghi, "An Approximation Algorithm for a Bottleneck Traveling Salesman Problem," Lecture Notes in Computer Science, pp. 223–235, 2006.

[14] C. Brucato, "The Traveling Salesman Problem," 2010.

[15] F. Greco, "Traveling Salesman Problem."

[16] M. Jünger, G. Rinaldi, and G. Reinelt, "The Traveling Salesman Problem," Handbooks Oper. Res. Manag. Sci., vol. 7, pp. 225–330, 1994.

[17] M. Held, A. J. Hoffman, E. L. Johnson, and P. Wolfe, "Aspects of the traveling salesman problem," IBM J. Res. Dev., vol. 28, no. 4, pp. 476–486, 1984.

[18] S. Tyagi and S. N. Singh, "Shortest path using neural network 1," vol. 8354, no. 2, pp. 112–119, 2013.

[19] J. J. Hopfield and D. W. Tank, "Biological Cybernetics 'Neural' Computation of Decisions in Optimization Problems," 1985.

[20] J. J. Hopfield, "Neurons with graded response have collective computational properties like those of two-state neurons.," Proc. Natl. Acad. Sci., 1984.

[21] D. S. Johnson, C. R. Aragon, L. A. Mcgeoch, C. Schevon, and R. Aragon, "Optimization Annealing : an Experimental Evaluation ;," Oper. Res., vol. 37, no. 6, pp. 865–892, 1989.

[22] [Y. J. Jeon, J. C. Kim, J. O. Kim, K. Y. Lee, and J. R. Shin, "An Efficient Simulated Annealing Algorithm for Network Reconfiguration in Large-Scale Distribution Systems," Power Eng. Rev. IEEE, vol. 22, no. 4, pp. 61–62, 2002.