Neural Comput & Applic (2015) 26:987–994
DOI 10.1007/s00521-014-1773-z

ORIGINAL ARTICLE

An Extended Self-Organizing Map based on 2-opt algorithm for solving symmetrical Traveling Salesperson Problem

Rashid Ahmad · DoHyeun Kim

Received: 6 September 2012 / Accepted: 30 November 2014 / Published online: 17 December 2014

Abstract Self-organizing map (SOM) is a powerful variant of neural network for solving optimization problems. Many researchers have reported SOM for Traveling Salesperson Problem; however, problems still exist due to the trapping of the optimization techniques at the local optimal position. In this work, we propose an Extended Self-Organizing Map based on 2-opt algorithm with one-dimensional neighborhood to approach the Symmetrical Traveling Salesperson Problem (STSP). We elaborate our approach for STSP where weights of neurons represent nodes that are placed in the polygonal domain. The selection of winner neuron of SOM has been extended to overcome the problem of trapping of SOM at local optima. The results of SOM are improved through 2-opt local optimization algorithm. We briefly discuss self-organization in neural networks, 2-opt algorithm, and extension applied to SOM. Finally, the algorithm is compared with Kohonen Self-Organizing Map and Evolutionary Algorithm. The results show that our approach performs better as compared to other techniques.

Keywords Artificial neural network · Self-Organizing Maps · Symmetrical Traveling Salesperson Problem · 2-opt Algorithm

R. Ahmad · D. Kim (✉)
Department of Computer Engineering, Jeju National University, Jeju-si, Jeju-do, Republic of Korea
e-mail: kimdh@jejunu.ac.kr

R. Ahmad
e-mail: Rashid141@gmail.com

1 Introduction

The traveling salesperson problem (TSP) is being studied from the last few decades. Given the list and pair-wise distance of cities, the task is to find the shortest tour that passes through each city exactly once. The problem seems to be very simple, but it has been reported as an NP-hard problem [1]. As a typical NP-hard problem, the classical TSP has attracted extensive research efforts in artificial intelligences methods [2]. This is because of its vast real-life applications, such as printed circuit-boards manufacturing [3, 4], data transmission in computer networks [5], power-distribution networks [6], image processing and pattern recognition [7], robot navigation [8], data partitioning [9], hole punching [10], vehicle routing [11], document locating, and so forth. Nowadays, the increasing diversity of applications requires increasingly large-scale TSP problems to be solved with precisions. The TSP can be stated as a search for a round trip of minimal total length visiting each node exactly once. Since there are $(N-1)!/2$ possible round trips for an N-city TSP, it is impractical to apply brute-force approach to the problem for a large N [12]. Many heuristic methods have been developed in order to achieve a near to optimal solution of the problem. The heuristics such as simulated annealing (SA) [13], genetic algorithm (GA) [14, 15], tabu search [16], neural networks [16] have demonstrated various degrees of strength and success to the TSP, among which neural networks are considered as a promising approach for large-scale TSP problems [16]. Due to the huge search space of a TSP ($n!$), it is not feasible to check all the solutions for city sets with many thousands of cities. Some engineering problems like VLSI designing need 1.2 million cities [17] with a limited time interval. A fast and effective heuristic method is thus needed to find the shortest route.

In this paper, we present an Extended SOM, a neural network-based algorithm improved with a 2-opt local optimization algorithm for solving symmetrical TSP. In our approach, the nodes of TSP are placed in a polygon domain and represented by weights of neuron in SOM Network. It is observed that the SOM could not reach to global optima and trapped at the local optima. To overcome this problem, we use 2-opt local optimization algorithm. The 2-opt algorithm locally optimizes the TSP tour, which leads to a global optima by the SOM. We called this approach as Extended SOM (ESOM).

The rest of the paper is organized as follows: In Sect. 2, the basic SOM is discussed. In Sect. 3, the extended SOM is discussed in detail for symmetrical TSP. Section 4 explains the experimentation and results. In Sect. 5, we describe the conclusion and future direction of the paper.

2 Related work

2.1 Basics of Self-Organizing Map

Teuvo Kohonen introduced a new type of neural network in 1975 that uses competitive, cooperative, and adaptive unsupervised learning [18]. The approach is based on winner takes most (WTM) and winner takes all (WTA) algorithms. WTA is a competitive learning algorithm. The distance to each neuron's synaptic weights is calculated when input vector is presented. The winner neuron is considered the one whose weights are most correlated with current input vector. The scalar product of input vector and considered synaptic weights are equal to correlation. The winning neurons then modify their synaptic weights with respect to the current input vector. However, the synaptic weights of other neurons remain the same. The learning process can be described by the following equation:

$$W_i = W_i + \eta(x - W_i) \tag{1}$$

where i $\in [0, 1, \ldots n]$, n represents the number of neurons, W_i is the synaptic weights of the winning neuron, η is learning rate and x stands for current input vector. This algorithm seems very simple and could be extended. The most common extension is giving more chance of winning to neurons that are rarely activated. Compared with WTA, the WTM strategy has better convergence. The main difference between WTA and WTM is that many neurons in WTM strategy adapt their synaptic weights in single learning iteration. Therefore, the winner neurons as well as the neighborhood neurons adapt their synaptic weights. The modification applied to the weights of neurons depends upon the distance of the neighborhood neurons from the winning neuron; the smaller the distance, the greater change will be applied to the weights of neighborhood neurons. This adaptation process can be described as:

$$W_i = W_i + \eta N(i, x)(w, W_i) \tag{2}$$

for all neurons i that belong to winner's neighborhood. W_i stands for synaptic weights of neuron i and x is current input vector. η stands for learning rate and $N(i,\ x)$ is a function that defines neighborhood. The SOM can be created when function $N(i,\ x)$ is defined as:

$$N(i, x) = \begin{cases} 1 & \text{for} \quad d(i, w) \le \lambda; \\ 0 & \text{for} \quad \text{others} . \end{cases} \tag{3}$$

where $d(i, w)$ is Euclidean distance between the ith and winning neuron and λ is neighborhood radius. For SOM training, the Euclidean distance between input vector and all neural weights has to be calculated. The distances of all neurons are compared, and the neuron that has the shortest distance to input vector is considered as a winner neuron. The weights of winner neuron are slightly modified to the direction represented by input vector. The weights of neighboring neurons are modified in the same direction based on their distance from the winner neuron. During each learning iteration, the λ and η are multiplied with $\delta\lambda$ and $\delta\eta$ respectively. These parameters are always > 0 and < 1. The λ and η will be updated during learning process and will become smaller during subsequent iterations. The SOM tries to organize itself globally during the beginning, and during the subsequent iterations, it performs local organization due to the smaller values of learning rate and neighborhood.

Figure 1 shows the classical Kohonen SOM. It maps input vectors of high dimension onto the map with one,

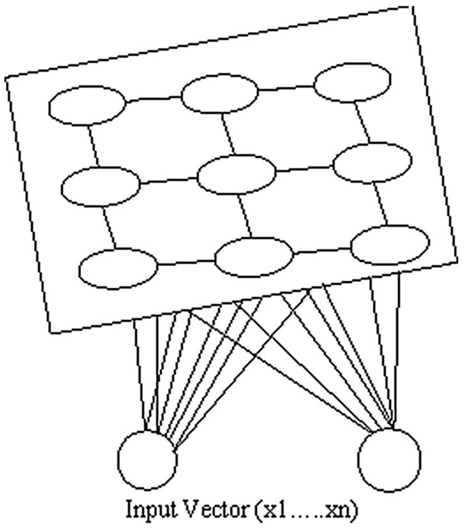


Fig. 1 Two-dimensional neighborhood SOM with input vector $(x_1 \ldots x_n)$

two, or more dimensions. The input patterns, which have the smaller Euclidean distance from one another in the input space, are put close to one another in the map. The input vector is passed through each neuron, and a matrix of output neurons is formed called SOM.

In one-dimensional neighborhood of matrix representation, every neuron has two neighbors, i.e., left and right as shown in Fig. 2.

In two-dimensional neighborhood of matrix representation, each neuron has four neighbors (left, right, top, and bottom). The classical two-dimensional neighborhood SOM with four neighbors is shown in Fig. 3. The neighborhood can be extended. Figure 4 shows the 8 neighbors scenario. The dimensions could be increased to as many as needed 1D, 2D, 3D, or more. However, 2D neighborhood is the most common.

Algorithm 1 shows the basic SOM proposed in [1]

```
INPUT:
N=number of neurons
I=number of iterations
OUTPUT:
Train_SOM
initialization;
i := 1;
while $i < N$ do
    $W_i \leftarrow rand()$ ;
end
for $i = 1 \rightarrow N$ do
    take one random input pattern;
    using Euclidean distance find the winning neuron;
    based on neighborhood function, find the neighbors;
    modify synaptic weights of these neurons;
    reduce $\eta$ and $\lambda$ ;
end
```

Algorithm 1: SOM training Algorithm

3 Extended SOM-based STSP solver

One-dimensional network must be created to solve TSP problem. The number of neurons and cities is equal. The weights of a neuron represent the coordinates of the city. In other words, a neuron and a city are assigned to each other and there exist a 1–1 mapping between the set of cities and set of neurons. All the neurons are organized in a vector. This vector represents sequence of cities that must be visited. However, some modifications need to be done before the SOM is able to fully solve this problem. This is because the real-valued neural weights may never equal

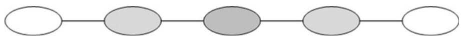


Fig. 2 One-dimensional neighborhood SOM

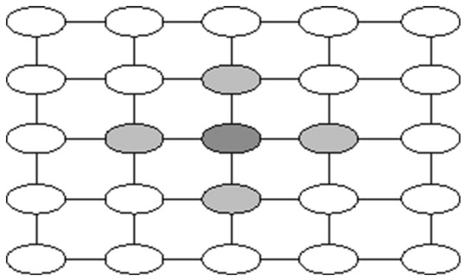


Fig. 3 Two-dimensional neighborhood SOM with four neighbors

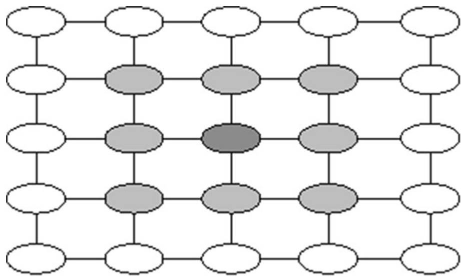


Fig. 4 Two-dimensional neighborhood SOM with eight neighbors

exactly the coordinates of the cities. To solve this problem, we use the algorithm proposed in [16] that will modify classical SOM solution and will map the neuron weights to the coordinates of the nodes/cities. Positions of cities and positions of neurons may not equal. However, adequate neural weights and city coordinates are very close to each other. An algorithm that modifies neural weights so they equal to city coordinates has also been applied. These weights need to be modified in such a way to restore the 1–1 mapping, assumed on the beginning. If a neuron a is assigned to city 'A,' it means that weights of neuron a are equal to coordinates of city 'A'.

Improvement in the learning of SOM network has been proposed in previous research in various ways [19, 20]. In this paper, we used a Gaussian-type neighborhood adaptation function $h(t, r)$, as used in [21].

$$h(t,r) = \frac{\alpha(1 - r^{*}f)}{\left[1 + \left(\frac{t}{\mathrm{cdenom}}\right)^{2}\right]} \tag{4}$$

This function decreases in both spatial and time domains. In the spatial domain, its value is the largest when node i is the winner node and it gradually decreases with increasing distance from i. Parameter α determines the initial value of $|h|$ while the parameter $f(0 < f < 1/r)$ determines the rate of decrease of $|h|$ in the spatial domain. In the time domain, t controls the value of $|h|$, whereas the parameter cdenom determines the rate of its decay.

Figure 5 shows the flow diagram of the SOM learning. The learning process continues for number of iterations

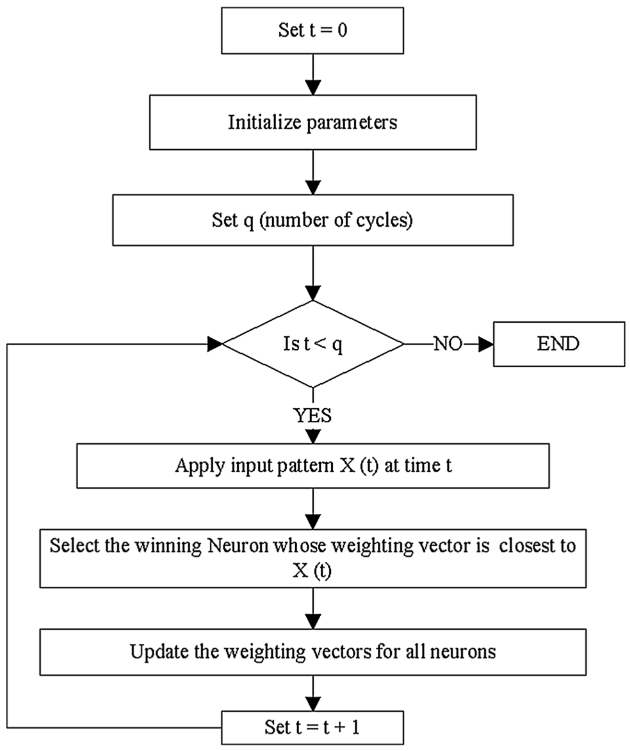

Set t = 0
Initialize parameters
Set q (number of cycles)
Is t < q
NO
END
YES
Apply input pattern X (t) at time t
Select the winning Neuron whose weighting vector is closest to X (t)
Update the weighting vectors for all neurons
Set t = t + 1

Fig. 5 Flow diagram of Self-Organizing Map learning

provided as input. The number of iterations can be set empirically to find the better results.

```
INPUT:
Weight set of neurons
Coordinates set
OUTPUT:
Mapped weights
initialization;
i=1;
while $i < N$ do
    nearest_city = nearest city to current neuron ;
    $i = i + 1$
    if nearest_city is not assigned to any neuron then
        assign nearest_city and current neuron;
    end
    else
        delete neuron[i];
    end
end
for $i = 1 \to N$ do
    if nearest_city is not assigned to any neuron then
        create a new_neuron and assign it to current city;
        nearest_neuron = find the nearest neuron to current
        city;
        insert new_neuron before or after nearest_neuron,
        depending on which tour is locally shorter;
    end
end
```

Algorithm 2: Mapping of Neuron and STSP cities

The algorithm 2 is used for mapping the real coordinates and neuron weights, which produce a good and fast solution; however, it is not locally optimal. We applied 2-opt algorithm, which is a well-known local optimization algorithm for producing locally as well as globally optimized solution. The solution obtained from SOM is good enough; therefore, the 2-opt will work fast even for large amount of cities. The 2-opt algorithm is based on one simple rule, and that is "remove the local loop and optimize locally". The algorithm works as follows:

- select part of tour
- reverse the tour
- insert back in the cycle
- test the results
- if the new cycle < the old tour
- replace the cycle else discard the new cycle
- Stop when no improvement can be done.

Suppose we have a cycle (A, B, C, D, E, F, A), we want to improve this cycle with 2-opt algorithm. Suppose we select path (B, C, D) and reverse it, the new cycle is (A, D, C, B, E, F, A). As shown in fig 6, after 2-opt optimization, the solution is locally optimal.

Figure 7 shows flow diagram of our proposed hybrid approach for solving STSP. Our approach is based on two approaches SOM & 2-opt as discussed above. The SOM is used for solving STSP; however, to overcome the shortcoming of SOM, we used 2-opt algorithm for better solution. The 2-opt reads the solution of SOM and finds the local optimal solution. The process continues until all the tours are visited.

4 Experimentation and results

We have carried out two types of experiments:

- Standard data set for TSP, i.e.,TSP Library (TSPLIB) [22].

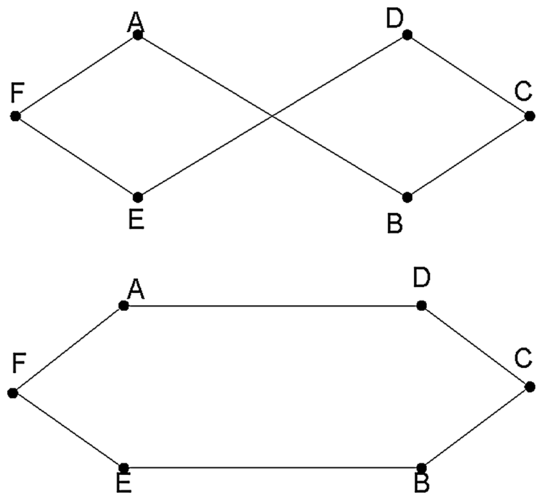


Fig. 6 Local optimization with 2-opt algorithm

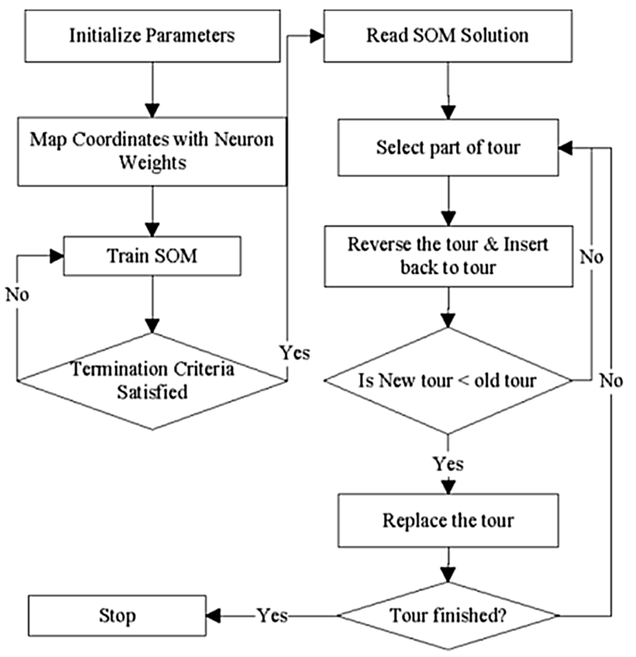


Fig. 7 Flow diagram of the proposed system

- Random cities.

TSPLIB data sets contain a number of local optimal values, which lead to trap the SOM-based system in the local optimal value; however, our two-layer hybrid approach does not trap in the local optimal value and finds the global optimal. The 2-opt local optimization algorithm helps to optimize the local tour, which could not be solved by SOM. Testing using randomly chosen cities is more objective. It is based on the Held-Karp Traveling Salesman bound [23]. An empirical relation for expected tour length is used:

$$L = k\sqrt{n \cdot R} \tag{5}$$

where L is expected tour length, n is a number of cities, R is an area of square box on which cities are placed and k is an empirical constant. For $n \geq 100$ it is:

$$k = 0.70805 + \frac{0.52229}{\sqrt{n}} + \frac{0.1.31572}{n} - \frac{0.3.07474}{n\sqrt{n}} \tag{6}$$

The selection of training parameter plays an important role during SOM learning. Optimal parameters should be selected adequately to a number of cities to achieve best results. We found the following best parameter values empirically during our experimentation.

- for 100 cities:

$\eta = 0.7$
$\Delta\eta = 0.987$
$\Delta\lambda = 0.999s$

- for 500 cities:

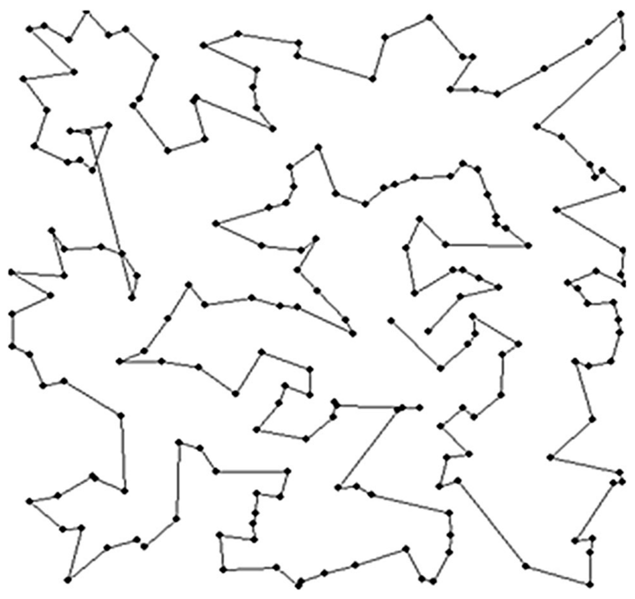


Fig. 8 SOM solution without 2-opt optimization. There are two local loops on the *left*

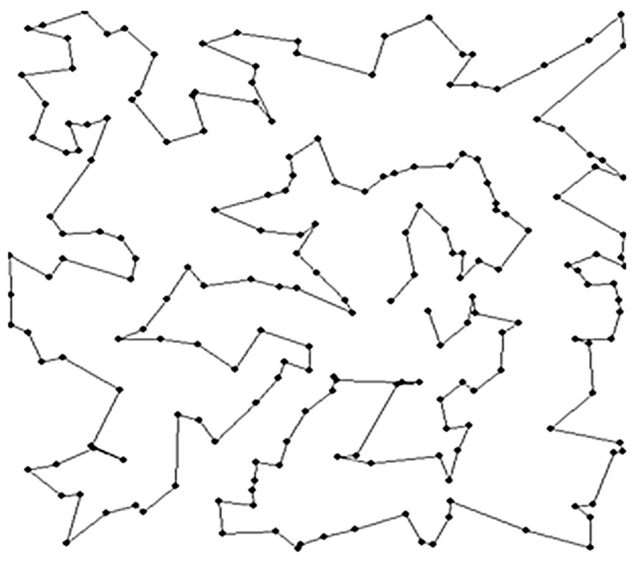


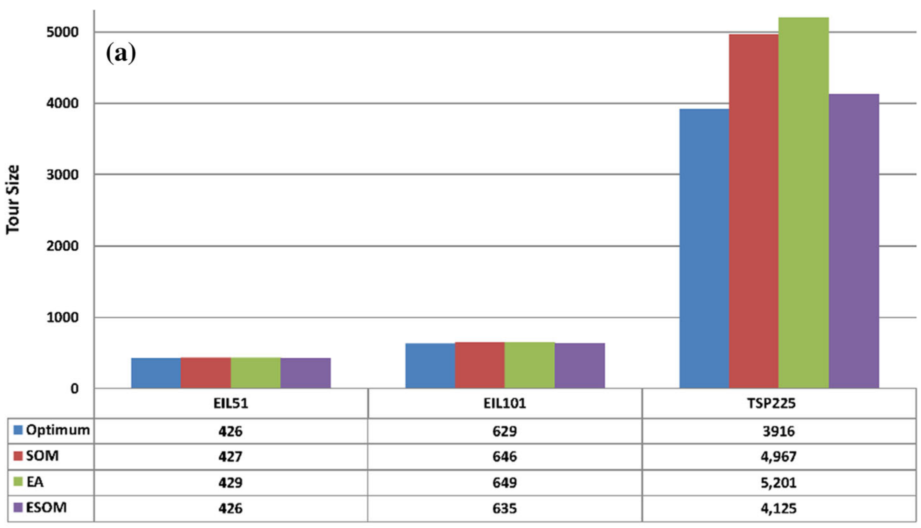
Fig. 9 SOM solution with 2-opt optimization. Loops on the *left* have been erased

$\eta = 0.8$
$\Delta\eta = 0.9985$
$\Delta\lambda = 0.994$

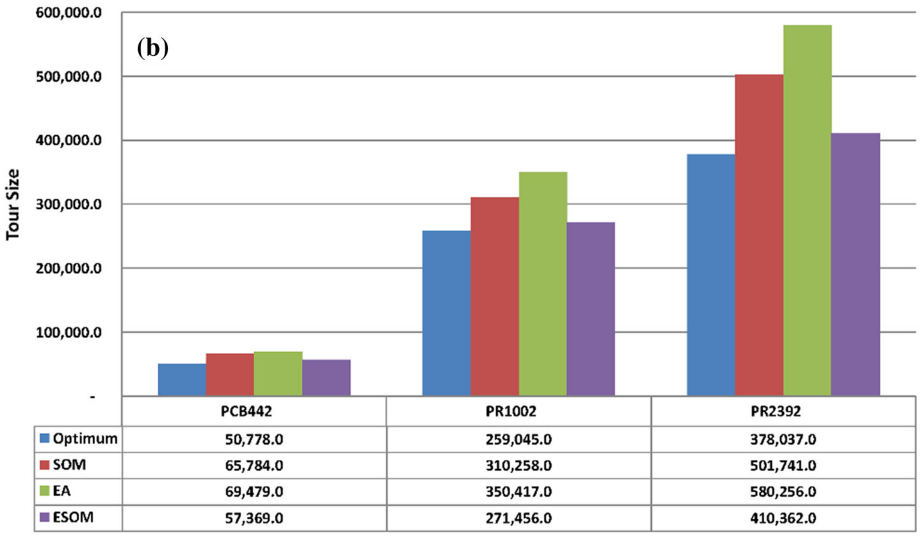
- for 1000 cities:

$\eta = 0.9$
$\Delta\eta = 0.9992$
$\Delta\lambda = 0.996$

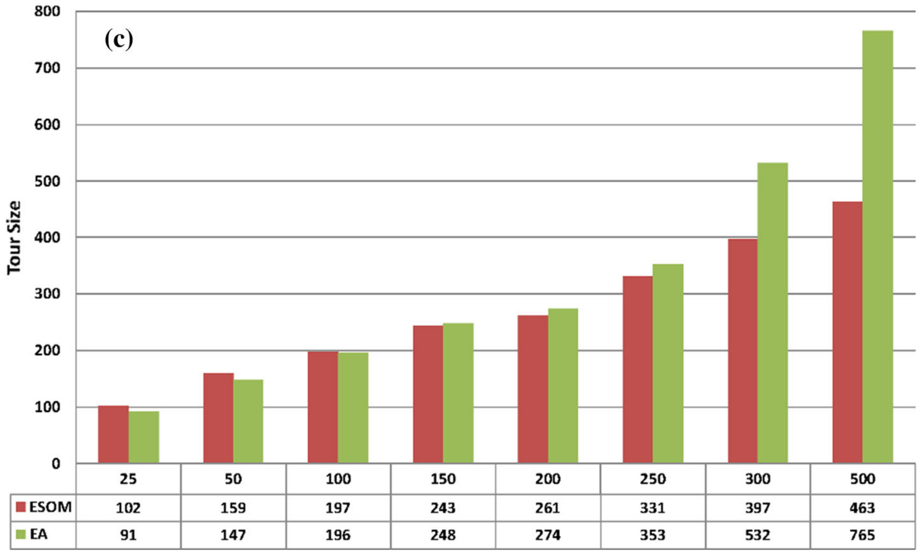
Fig. 10 Comparison of results **a** comparison of ESOM with SOM and EA on TSPLIB, **b** comparison of ESOM with SOM and EA on TSPLIB, **c** comparison of ESOM with EA on random data sets



	EIL51	EIL101	TSP225
🟦 Optimum	426	629	3916
🟥 SOM	427	646	4,967
🟩 EA	429	649	5,201
🟪 ESOM	426	635	4,125



	PCB442	PR1002	PR2392
🟦 Optimum	50,778.0	259,045.0	378,037.0
🟥 SOM	65,784.0	310,258.0	501,741.0
🟩 EA	69,479.0	350,417.0	580,256.0
🟪 ESOM	57,369.0	271,456.0	410,362.0



	25	50	100	150	200	250	300	500
🟥 ESOM	102	159	197	243	261	331	397	463
🟩 EA	91	147	196	248	274	353	532	765

Table 1 ESOM comparison with SOM[16] and evolutionary algorithm

Instances	Optimum	SOM		EA		ESOM	
		Ave. result	Best result	Ave. result	Best result	Ave. result	Best result
EIL51	426	431	427	434	429	429.2	426.2
EIL101	629	662	646	667.3	649	638.3	635
TSP225	3,916	5,364	4,967	5,936	5,201	4,432	4,125
PCB442	50,778	69,526	65,784	73,213	69,479	66,932	57,369
PR1002	259,045	354,896	310,258	390,245	350,417	299,685	271,456
PR2392	378,037	587,023	501,741	612,302	580,256	450,216	410,362

The numbers of iterations were set to 20,000 for all the cases.

All computations were performed on AMD Athlon (tm)64-bit X2 Dual Core 6000+ 3.10 GHz processor with 4GB Physical Memory.

Six different size data sets were selected for testing our proposed approach, i.e., EIL51, EIL101, TSP225, PCB442, PR1002, and PR2392. The SOM solution for a 200 random cities problem without 2-opt optimization which resulted in two loops in the resulted solution as shown in Fig. 8. The 2-opt local optimization identify and replace the loops in the SOM resulted solution with optimal results as shown in Fig. 9. Figure 10a shows the comparison of EA, SOM, and ESOM on data sets EIL51, EIL101, and TSP225. It is clear from the figure that the tour size is almost the same in small size data set; however, when the number of cities grows, the tour size and difference of tour size increase. Figure 10b shows the comparison of EA, SOM, and ESOM on data sets PCB442, PR1002, and PR2392. It is very clear from Fig. 10b that the ESOM performs better than and near optimal than the other two counterparts.

Experiments were also carried out on different size random data sets 50, 100, 150, 200, 250, 300, and 500 cities, respectively. Rectangular box edge length was set to 500. The ESOM was run for 50 times on each city set, and the statistics were generated. It has been observed that SOM takes relatively shorter time to generate an optimal tour. During our experimentation, the 1000 city set is solved in less than 2.5 s. ESOM approach generates solutions that are almost 10 % better than the other evolutionary techniques and classical SOM. However, in most of the cases, the difference is very small, i.e., just a few percent. EA used enhanced edge recombination (EER) operator [24, 25], survival of the fittest (where always the worst solution is replaced), and tournament parent selection with tournament size depending on number of cities and population size. Mixed along with self-adapting mutation rate has been used. The mutation rate is different for every genotype, and the modification is similar to that of evolution strategies. Due to this adaptable strategy for mutation rate, it is not needed to check manually which parameters are optimal for each city set. The convergence of population is the stopping criteria. Population size was set to 1000 (as in [25]). With smaller populations, EA did not work that well. When EA stopped, its best solution was optimized by 2-opt algorithm. The ESOM is run for 10 times for each city set, and the statistics were calculated. For data sets, EIL51, EIL101, and the EA run for 10 times. For other experimental sets, the EA was run only once. The optimum solutions of city sets are taken from TSPLIB.

Experimental results show that EA and SOM perform better in small city sets. Both the average and best results of EA are near ESOM's results. For city sets with 50 or less cities, EA finds optimum in almost every execution. Results for above 100 cities are comparable for both algorithms. As shown in Table 1, the ESOM performs better than EA and SOM for TSPLIB data sets. The difference between the two algorithms increases as the number of instances increases in a set. With more cities, search space increases significantly and EA needs bigger population size, while SOM stuck in local optimal value. The EA needs larger population size and larger generation size for more instances, which lead to poor performance both in the form of time and optimality. The EA algorithm is much slower, while SOM produces the results faster than ESOM.

5 Conclusion

It seems that SOM 2-opt hybrid is a powerful algorithm for the TSP. It outperforms both EA and SOM algorithms. Its speed might not be impressive as compared to SOM, but it is more effective in terms of optimality.

In order to improve the efficiency of ESOM, there are couple of things that can be optimized. Here are some of them:

- An optimal network parameter settings should be found ($\eta, \Delta\eta, \Delta\lambda, number\ of\ iterations$)
- Experiments with other self-organizing networks should be performed; Gaussian neighborhood and conscience mechanism may be applied. Conscience mechanism can improve TSP solutions generated by neural networks, as reported in [26].
- 2-opt algorithm is not very sophisticated. Some other optimization method may improve the results.

There are many algorithms that solve permutation problems. Evolutionary algorithms have many different operators that work with permutations. EER is one of the best operators for the TSP [25]. However, it was proved that other permutation operators, which are worse for the TSP than EER, are actually better for other permutation problems (like warehouse/shipping scheduling) [25]. Therefore, it might be possible that SOM 2-opt hybrid might work better for other permutation problems than for the TSP.

Acknowledgments This work was supported by the Industrial Strategic Technology Development Program (No. 10043907, Development of high performance IoT device and Open Platform with Intelligent Software), and this research was supported by the MSIP (Ministry of Science, ICT and Future Planning), Korea, under the ITRC (Information Technology Research Center) support program (NIPA-2014-H0301-14-1048) supervised by the NIPA (National IT Industry Promotion Agency).

References

1. Garey Michael R, Johnson David S (1979) Computers and intractability: a guide to the theory of NP-completeness. W. H. Freeman & Co., New York
2. Lawler EL, Lenstra JK (1985) The Traveling Salesman Problem-guided tour of combinatorial optimization. Wiley, New York
3. Fujimura K, Obu-Cann K, Tokutaka H (1999) Optimization of surface component mounting on the printed circuit board using SOM TSP method. In: Proceedings of the 6th ICONIP, vol 1, pp 131–136
4. Fujimura K, Fujiwaki S, Kwaw OC, toktaka H (2001) Optimization of electronic chip-mounting machine using SOM TSP method with 5 dimensional data. Proc ICII 4:26–31
5. Ali MM, Kamoun F (1993) Neural networks for shortest path computation and routing in computer networks. IEEE Trans Neural Netw 4(6):941–954
6. Onoyama T, Maekawa T, Kubota S, Taniguchi Y, Tsuruta S (2002) Intelligent evolutional algorithm for distribution network optimization. In: Proceedings of international conference control applications, pp 802-807
7. Banasza KD, Dale GA, Watkins AN, Jordan JD (1999) An optical technique for detecting fatigue cracks in aerospace structures. In: Proceedings of the 18th ICIASF, pp 1–27
8. Barrel D, Perrin JP, Dombre E, Liengeois A (1999) An evolutionary simulated annealing algorithm for optimizing robotic task ordering. In: Proceedings of IEEE ISATP, pp 157–162
9. Cheng CH, Lee WK, Wong KF (2002) A genetic algorithm-based clustering approach for database partitioning. IEEE Trans Syst Man Cybern C Appl Rev 32(3):215–230
10. Ascheuer N, unger MJ, Reinelt G (2000) A branch and cut algorithm for the asymmetric Traveling Salesman Problem with precedence constraints. Comput Optim Appl 17(1):61–84
11. Laporte G (1992) The vehicle routing problem: an overview of exact and approximate algorithms. Eur J Oper Res 59:345–358
12. WangWei Artificial (1995) Neural network theory applications. Beijing University of Aeronautic and Astronautic Science and Technology Press, Beijing
13. Kirk SG, Jr Patrick, Gelatt CD, Vecchi MP (1983) Optimization by simulated annealing. Science 220:671–680
14. Goldberg DE (1989) Genetic algorithms in search, optimization and machine learning. Addison-Wesley, Reading
15. Yan XS, Liu HM, Yan et al (2007) A fast evolutionary algorithm for traveling salesman problem. In: Proceedings of the third international conference on natural computation, vol 4, pp 85–90
16. Brocki L, Korinek D (2007) Kohonen self-organizing map for the Traveling Salesperson Problem. In: Recent advances in mechatronics. Springer, Heidelberg, pp 116–119. doi:10.1007/978-3-540-73956-2_24
17. Korte B (1989) Applications of combinatorial optimization. In: Mathematical programming, Volume 6 of Math. Appl. (Japanese Ser.). SCIPRESS, Tokyo, pp 1–55
18. Kohonen T (2001) Self-organizing maps, Springer series in information sciences, vol 30, 3rd edn. Springer, Berlin
19. Xu X, Tsai WT (1991) Effective neural algorithms for the Traveling Salesman Problem. Int J Neural Netw 4(2):193–205
20. Lin S, Kernighan BW (1973) An effective heuristic algorithm for the Traveling-Salesman Problem. Oper Res 21:498–516
21. Mitra S, Pal SK (1994) Self-organizing neural network as a fuzzy classifier. Syst IEEE Trans Man Cybern 24(3):385–399
22. Reinelt G (1995) TSPLIB 95 documentation. University of Heidelberg, Heidelberg, Germany. http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/
23. Johnson DS, McGeoch ALA, Rothberg TAEE (1996) Asymptotic experimental analysis for the Held-Karp traveling salesman bound. In: Proceedings of the seventh annual ACM-SIAM symposium on discrete algorithms, Atlanta, Georgia, USA, pp 341–350
24. Michalewicz Z (1996) Genetic algorithms data structures evolution programs. Springer, Berlin
25. Starkweather T, McDaniel S, Mathias K, Whitley D, Whitley C (1991) A comparison of genetic sequencing operators. In: Proceedings of the 4th international conference on genetic algorithms, San Mateo
26. Burke Laura I (1994) Neural methods for the Traveling Salesman Problem: insights from operations research. Int J Neural Netw 7(4):681–690