

A Kohonen-like Decomposition Method for the Traveling Salesman Problem—KNIES_DECOMPOSE

Necati Aras¹, İ. Kuban Altınel² and John Oommen³

Abstract. In addition to the classical heuristic algorithms of operations research there have also been several approaches based on artificial neural networks which solve the traveling salesman problem (TSP). Their efficiency, however, decreases as the problem size (number of cities) increases. An idea to reduce the complexity of a large-scale TSP instance is to decompose or partition it into smaller subproblems, which are easier to solve. In this paper we introduce an all-neural decomposition heuristic that is based on a recent self-organizing map called KNIES which has been successfully implemented in solving both the Euclidean TSP and the Euclidean Hamiltonian path problem.

1 INTRODUCTION

Of all the families of neural networks described in the literature, the Kohonen self-organizing neural network has been the most widely investigated one [1]. This is not surprising, given its simplicity and the wide variety of problems to which it may be applied. One of these problems is the Euclidean traveling salesman problem (TSP) from operations research and there have been many attempts to solve it by self-organizing maps (SOM) [2, 3]. Any algorithm devised to solve the TSP tries to answer the following question: Given a set of N cities and distances for each pair of cities, what is the shortest tour that visits each city exactly once?

Although SOM is very powerful, there is a relatively vast amount of information that it ignores in the training process. This information, which can be informally perceived as the global information, is the statistical information resident in the data points (input vectors) in their entirety. Thus, although in the SOM the neurons asymptotically learn the distribution of the points statistically, the SOM does so only by virtue of the points themselves and not by utilizing the information resident in the overall set of points, such as their mean etc.

The decomposition into subproblems is a known approach for solving large instances of TSPs. It is easier to solve the subproblems because the size of each subproblem is much smaller. Once the solutions to each subproblem are obtained they can be combined to approximate the solution of the original problem.

In this paper a new method called KNIES_DECOMPOSE is introduced. The new method is based on a recent self-organizing map called The Kohonen Network Incorporating Explicit Statistics (KNIES) [4]. KNIES has already been successfully implemented to

solve both the TSP (KNIES_TSP) [5] and the Euclidean Hamiltonian path problem (KNIES_HPP) [6]. The primary difference between the SOM and the KNIES is the fact that every iteration in the training phase includes two distinct modules—the *attracting* module and the *dispersing* module. In the attracting module a subset of the neurons migrate towards the data point that has been presented to the network. This phase is essentially identical to the learning phase of the SOM. However, subsequent to this phase the rest of the neurons which have not been involved in the attracting module participate in a dispersing (repellent) migration. Indeed, these neurons now move away from their current positions in a manner that ensures that the global statistical properties of the data points are imitated by the neurons. Thus, although in the SOM the neurons *asymptotically* find their places both statistically and topologically, in the KNIES they collectively maintain their means to be the means of the data points which they represent.

2 DECOMPOSITION APPROACH TO THE EUCLIDEAN TSP

An idea to reduce the complexity of a large-scale traveling salesman problem instance is to decompose or partition it into smaller subproblems, which are easier to solve. The partitioning is performed by clustering the cities of the original problem in a way that structural properties of the problem instance are preserved. Generally speaking, the problem of size n is divided into k nonoverlapping clusters C_i of size n_i ($C_i \cap C_j = \emptyset \ \forall i \neq j$), where $\max \{n_i : 1 = 1, \dots, k\} \ll n$.

Clustering is a research topic itself and there are numerous mathematical methods [7]. After the cities are partitioned into different clusters, it would be possible to proceed in two different ways. One way is to solve the traveling salesman subproblem for each cluster and then to join the subtours to form a global tour. The traveling salesman subproblems can be solved using any efficient TSP heuristic. For the generation of the global tour the following procedure might be applied: We begin with an arbitrary subtour T_1 . This subtour is then connected to another subtour T_2 to form a new tour. In general, a subtour T_i is connected to the global tour \hat{T}_i generated so far by removing one edge of T_i and one edge of \hat{T}_i and then by replacing them by two new edges connecting T_i and \hat{T}_i to form a new tour \hat{T}_{i+1} . This procedure, however, has an important disadvantage: the subtours are not optimized with respect to the edges connecting the clusters. So a better approach is to obtain the global tour *before* the subproblems are solved in order to integrate the obtained information into the local TSP-heuristic. In this way the local solution also considers the city n_{i_e} where the global tour enters the cluster, and the city n_{i_l} , where it leaves the cluster and so takes into account the outline of the global tour. As a result better global solutions are

¹ İ.d.e.a. Training, Consultancy and Research Inc., İstanbul, Turkey, email: necatia@idea.com.tr

² Industrial Engineering Department, Boğaziçi University, İstanbul, Turkey, email: altinel@boun.edu.tr

³ School of Computer Science, Carleton University, Ottawa, Canada, email: oommen@scs.carleton.ca

obtained. This latter approach is the second way of proceeding after the cities are partitioned. We give the steps of this approach, which we will adopt for KNIES_DECOMPOSE in the following.

1. Compute the centroid or the mean of the cities in each cluster.
2. For every cluster determine the r nearest clusters in regard to the centroid.
3. Compute the convex hull of each cluster.
4. Compute the exact distances for the r nearest clusters (as the shortest distance between the nodes of the convex hulls).
5. Obtain a global tour through the clusters by a TSP-heuristic using exact distances, if available, or otherwise the distances between the means which gives an entering and a exiting city for each cluster.
6. Apply a suitable heuristic to find the Hamiltonian path in every cluster connecting the entering to the exiting city visiting all the cities in that cluster.
7. Merge the edges between the exiting city of one cluster and entering city of the subsequent cluster in the global tour and the Hamiltonian paths to form a tour for the original problem.

3 AN ALL-NEURAL DECOMPOSITION APPROACH: KNIES_DECOMPOSE

The first step of KNIES_DECOMPOSE is to partition the cities into clusters. The clustering is accomplished using vector quantization. Here the number of the clusters is a parameter and there are as many codebook vectors as the number of clusters. The codebook vectors are moved in the two-dimensional space until they find their final places and then the closest codebook vector is found for each city. Hence the input space is divided into clusters each of which is represented by a codebook vector. At this point the codebook vectors and the mean of the cities in a given cluster (the centroid of the cluster) coincide. Therefore the codebook vectors representing the clusters can be used to find the global tour through the clusters. However, in order to obtain a better discrimination of the clusters (i.e., near-optimal cluster boundaries) we make use of the *intra*regional and *inter*regional polarizing.

The aim of intra-regional polarizing is to represent each cluster C_k by a number of codebook vectors M_k where M_k increases with the number of cities located in that cluster which is denoted as N_k . Specifically, $M_k = \lfloor 0.3 N_k \rfloor$. If there are three or less cities in a cluster, then M_k is set equal to one. The set of codebook vectors for cluster k , $\{Q_{k,j} : 1 \leq j \leq M_k\}$ are initially located on a circle the center of which is the mean of the cities belonging to that cluster. Each city $P_{k,i}$ in that cluster is then presented to the network, and the location of nearest codebook vector $Q_{k,j}$ is updated according to the formula given below:

$$Q_{k,j}(t+1) = (1 - \alpha(t)) Q_{k,j}(t) + \alpha(t) P_{k,i} \quad (1)$$

Other codebook vectors in that cluster maintain their positions. $\alpha(t)$ is decremented linearly from unity for the initial learning phase and then switched to 0.2 and is decreased linearly for the fine-tuning phase. After the individual clusters have been represented by M_k codebook vectors they are tested to see whether they adequately classify the cities within their clusters. Therefore, the inter-regional polarizing phase has been employed where the codebook vectors do not find their places by learning only from the cities within their own clusters (as in the intra-regional polarizing phase) but they are also migrated in such a way that they polarize away from the cities of

the neighboring clusters. The principle by which this is done is as follows:

Suppose that a point $P \in C_k$ is examined. Also assume that the two closest codebook vectors to P (among all the codebook vectors) are Q_a and Q_b . If both Q_a and Q_b do not belong to the cluster C_k , clearly, the information content in P (with respect to Q_a and Q_b) is misleading, and so it is futile to migrate Q_a and Q_b using this information. However, if both of them are intended to represent C_k , clearly, the information in P can be used to achieve an even finer tuning to their locations. Thus, in this scenario, both Q_a and Q_b are moved marginally from their current locations along the line towards P . The final scenario is the case when one of them, Q_a (Q_b), correctly belongs to C_k , and the other, Q_b (Q_a), belongs to a different partition. In this case, the information in P can be used to achieve an even finer tuning to their locations by migrating Q_a (Q_b) marginally from its current location along the hyperline towards P , and migrating the other codebook vector Q_b (Q_a) marginally from its current locations along the line away from P . Since we do not want the “straggler” points (the points which are misclassified, but which probably would not have been correctly classified even by an optimal classifier) to completely dictate (and thus, disturb) the polarizing, this migration is invoked only if the node P lies within a pre-specified window of interest, W . This restriction has also been recommended in the literature [1], and typically, this window, W , is a hypersphere centered at the bisector between the codebook vectors Q_a and Q_b . Also, as recommended in the literature, the polarizing of both Q_a and Q_b (when both of them correctly classify P) is made to be of much smaller magnitude than in the scenario when either of them misclassifies it. These steps are formally given below:

$$\begin{aligned} Q_a(t+1) &= (1 - \epsilon\gamma) Q_a(t) + \epsilon\gamma P & \text{if } Q_a, Q_b \in C_k \\ Q_b(t+1) &= (1 - \epsilon\gamma) Q_b(t) + \epsilon\gamma P & \text{if } Q_a, Q_b \in C_k \\ Q_a(t+1) &= (1 - \gamma) Q_a(t) + \gamma P & \text{if } Q_a \in C_k; Q_b \in C_j \\ Q_b(t+1) &= (1 + \gamma) Q_b(t) - \gamma P & \text{if } Q_a \in C_k; Q_b \in C_j \\ Q_a(t+1) &= (1 + \gamma) Q_a(t) - \gamma P & \text{if } Q_a \in C_j; Q_b \in C_k \\ Q_b(t+1) &= (1 - \gamma) Q_b(t) + \gamma P & \text{if } Q_a \in C_j; Q_b \in C_k \\ Q_a(t+1) &= Q_a(t) & \text{if } P \notin W \\ Q_b(t+1) &= Q_b(t) & \text{if } P \notin W \end{aligned} \quad (2)$$

There are three parameters in these update equations; γ , ϵ , and the diameter of the hypersphere W centered at the bisector of the two nearest codebook vectors. Except ϵ , which is kept constant at 0.25, experiments were performed with different values of parameters γ and the diameter of the hypersphere W in order to see the effect of the inter-regional polarizing on the quality of the solution (i.e., tour length). γ assumed values in the interval $(0, 1)$ with increments of 0.1 and the diameter of W was set equal to some percentage of the distance between the two codebook vectors. Again, we experimented with different values of percentages.

The output of the inter-regional polarizing phase is the final partitioning of the cities into clusters. The next step is to determine the global tour through the clusters. To accomplish this, the centroid of each cluster is found by computing the coordinatewise mean of the cities located in that cluster and the algorithm KNIES_TSP_Global is invoked. KNIES_TSP_Global is a modified version of KNIES_TSP and quickly yields a tour passing through the centroids. Hence, the sequence in which the global tour visits the clusters is determined. The difference between the two is explained in [4]. The next step is to find out the entering and exiting cities for each cluster. This is done as follows. For two subsequent clusters in the global tour the two nearest cities is found such that one city belongs to the first

cluster and the other one belongs to the second cluster. These two cities constitute the bridge between the two clusters, in other words the global tour passes from one cluster to another through these two cities.

After the entering and exiting cities for each cluster are found, the remaining task becomes the determination of the Hamiltonian path between these cities which is done by running the algorithm KNIES_HPP_Global which is a modified version of KNIES_HPP [6]. When the Hamiltonian paths for each cluster are glued together, the final tour is obtained. The decomposition approach is tested for some instances obtained from TSPLIB.

4 COMPUTATIONAL RESULTS

The crucial parameter for the decomposition approach is the number of clusters since the parameters of both the algorithms KNIES_TSP_Global and KNIES_HPP_Global are fixed. Particularly, the following values are used for the parameters of KNIES_TSP_Global that is used to determine the order in which the clusters are visited: $K_\sigma = 0.8$, $\omega = 0.2$, and $\sigma = 20$ initially. Here, σ is the parameter that adjusts the strength with which the neurons in the neighborhood of the winning neuron are pulled towards the city presented to the network. K_σ is used to decrease the value of σ at each epoch, so that the convergence of all the neurons to some position is guaranteed. ω plays an important role in both KNIES_TSP_Global and KNIES_HPP_Global by determining the number of neurons that participate in the attracting and the dispersing modules of KNIES. $\omega = 0.2$ means for example that 40 per cent of the neurons are used for the attracting module being 20 per cent on one side and the other 20 per cent on the other side of the winning neuron when the neurons are indexed on a virtual elastic band.

Furthermore, the initial number of neurons is set equal to the number of clusters. The same parameter setting is also adopted for KNIES_HPP_Global where $K_\sigma = 0.8$, $\omega = 0.2$, and $\sigma = 20$ initially. The initial number of neurons is set equal to 0.3 times the number of cities (including the entering and exiting cities) in the cluster for which the Hamiltonian path is to be found. If the number of neurons turns out to be less than three, then the algorithm is run with three neurons initially. These values for the parameters are selected since the algorithms KNIES_TSP_Global and KNIES_HPP_Global provided satisfactory results with these parameters for most of the instances.

The steps of KNIES_DECOMPOSE are illustrated in the following figures while solving the instance *e11101* for 10 clusters. Figure 1 (a) shows the result of the partitioning. The cities are divided into 10 clusters. The darker dots represent the centroids of the clusters. Figure 1 (b) contains the global tour through the clusters computed by using KNIES_TSP_Global with the cluster centroids as the input. The global tour gives the order in which clusters are visited on the final tour passing through all the cities in the instance. Hence, it is now possible to determine the entering and exiting cities in each cluster. The exiting city of a cluster and the entering city of the subsequent cluster are connected and this connection constitutes the bridge between the clusters (according to the order in which clusters are visited). This can be seen in Figure 1 (c). Figure 1 (d), Figure 2(a), and (b) are snapshots taken after the Hamiltonian paths are determined by KNIES_HPP_Global for one, two, and three clusters, respectively, and glued together with the bridges between these clusters. As the Hamiltonian path is found for each cluster and glued with that of the subsequent cluster, the TSP tour for the original problem grows

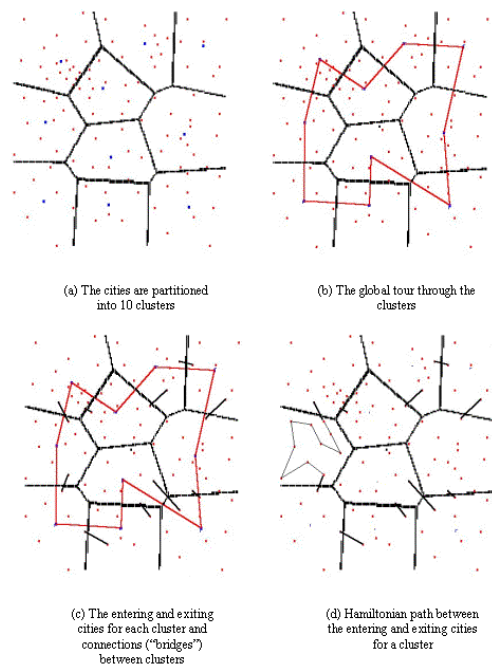


Figure 1. All-neural decomposition approach.

and KNIES_TSP_Global for the same instances are given in Table 1.

Table 1. Comparison of the results obtained for different algorithms instances.

Instance	KNIES DECOMP. (clusters)	KNIES_TSP ($M, \sigma, K_\sigma, \omega$)	KNIES_TSP Global ($M, \sigma, K_\sigma, \omega$)
att532	29388.9 (13)	29551.6 (200,30,0.8,0.15)	29569.8 (400,45,0.8,0.1)
bier127	126080.8 (9)	121548.7 (100,15,0.8,0.1)	121923.7 (125,50,0.8,0.1)
eil51	440.9 (5)	438.2 (10,25,0.8,0.1)	438.2 (20,50,0.8,0.05)
eil76	572.9 (6)	564.8 (90,20,0.8,0.15)	567.5 (15,5,0.8,0.2)
eil101	672.0 (4)	658.3 (20,25,0.8,0.2)	664.4 (40,35,0.8,0.25)
kroA200	30184.9 (12)	30200.8 (200,25,0.8,0.25)	30444.9 (160,10,0.8,0.05)
lin105	14693.1 (8)	14664.4 (100,50,0.8,0.2)	14564.6 (100,35,0.8,0.05)
pcb442	54838.6 (3)	56399.9 (500,30,0.8,0.1)	56082.9 (450,40,0.8,0.15)
pr107	49103.9 (4)	44628.3 (100,45,0.8,0.1)	44491.1 (60,25,0.8,0.25)
pr124	60931.5 (3)	59075.7 (25,25,0.8,0.1)	59320.6 (125,10,0.8,0.05)
pr136	98641.2 (8)	101156.8 (75,40,0.8,0.15)	101752.4 (30,35,0.8,0.05)
pr152	76072.1 (15)	74395.5 (180,30,0.8,0.05)	74629.0 (60,10,0.8,0.2)
rat195	2517.0 (2)	2607.3 (200,25,0.8,0.25)	2599.8 (200,25,0.8,0.1)
rd100	8296.9 (9)	8075.7 (80,20,0.8,0.25)	8117.4 (60,10,0.8,0.05)
st70	699.8 (3)	685.2 (40,10,0.8,0.05)	690.7 (30,45,0.8,0.05)

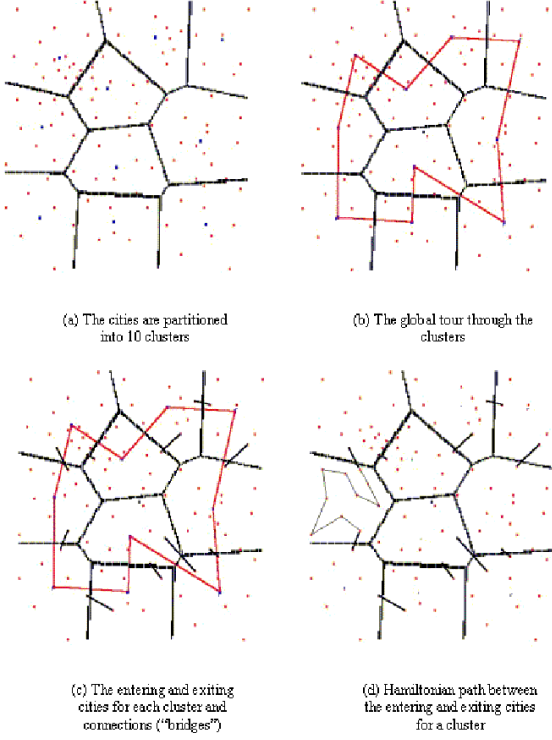


Figure 2. All-neural decomposition approach (continued).

gradually and gets its final shape given in Figure 2 (c). The final TSP tour for the instance `eil101` can be seen in Figure 2 (d) without the cluster borders.

For each instance, experiments are performed with different number of clusters so that on the average no more than 50 and no less than 14 cities belong to each cluster which is achieved when the following is satisfied: $\lfloor 0.02 N \rfloor \leq \text{number of clusters} \leq \lfloor 0.07 N \rfloor$. Here N is the problem size, i.e., the number of cities. This is because when the number of cities in any cluster exceeds 50 or so, then the computational time of determining the Hamiltonian path for that cluster increases. Hence, the number of clusters should not be less than $\lfloor 0.02 N \rfloor$. On the other hand, when the cities are divided into too many clusters, then the number of cities in each cluster gets smaller, and the preclustered structures are destroyed. Thus we move away from the global optimal. Thus, the number of clusters has to be less than $\lfloor 0.07 N \rfloor$. To give an example, the number of clusters varies between $\lfloor 0.02 \times 532 \rfloor = 10$ and $\lfloor 0.07 \times 532 \rfloor = 37$ for `att532`. As it is pointed out in the previous section the parameters of the interregional polarizing assumed the following values: $\epsilon = 0.25$, and both γ and the diameter of W range between zero and one. The overall best result is achieved when the cities are divided into 13 cities.

The overall best results obtained by KNIES_DECOMPOSE for different TSP instances and the best values provided by KNIES_TSP

Table 2 contains the relative deviations of the three approaches from the optimal values. As it can be observed in the table the success of KNIES_DECOMPOSE increases as the problem size increases. For `att532`, `pcb442`, `kroA200`, `rat195`, KNIES_DECOMPOSE provides shorter tour lengths than both KNIES_TSP and KNIES_TSP_Global. If only these four instances are considered, the average relative deviations from the optimal tour lengths become 7.03, 8.94, and 8.93 for the KNIES_DECOMPOSE, KNIES_TSP, and KNIES_TSP_Global, respectively. The most important advantage of KNIES_DECOMPOSE which cannot be perceived in the table is its speed. Compared with the other two approaches it is possible to obtain solutions very quickly even for large problems. This is due to the fact that once the sequence of the clusters on the global tour and subsequently the entering and the exiting cities for each cluster are determined, the Hamiltonian paths between these cities within the clusters may be found simultaneously. It is even possible to distribute the solution of the Hamiltonian path problem on different computers.

5 ACKNOWLEDGEMENTS

The computational experiments of this research were realized on the hardware donated to the Department of Industrial Engineering at Boğaziçi University by Hewlett - Packard. The first and second authors are partially supported by the Boğaziçi Research Fund Grant 98A301D. The third author is partially supported by the National Science and Engineering Research Council of Canada (NSERC).

Table 2. Relative deviations from the optimal tour length (per cent)

Instance	Opt.	KNIES	KNIES	KNIES
	Value	DECOMP.	TSP	TSP.Glo.
att532	27686	6.15	6.74	6.80
bier127	118282	6.59	2.76	3.08
eil51	426	3.49	2.86	2.86
eil76	538	6.49	4.98	5.48
eil101	629	6.84	4.66	5.63
kroA200	28568	5.66	5.72	6.57
lin105	14379	2.18	1.98	1.29
pcb442	50778	7.99	11.07	10.44
pr107	44303	10.84	0.73	0.42
pr124	59030	3.22	0.08	0.49
pr136	96772	1.93	4.53	5.15
pr152	73682	3.24	0.97	1.29
rat195	2323	8.35	12.24	11.92
rd100	7910	4.89	2.09	2.62
st70	675	3.67	1.51	2.33
Avg. Rel. Deviations		5.41	4.19	4.42

REFERENCES

- [1] T. Kohonen, *Self-Organizing Maps*, Springer-Verlag, Berlin, Germany, 1995.
- [2] J. Y. Potvin, 'The Traveling Salesman Problem: A Neural Network Perspective', *ORSA Journal on Computing*, **5**, 328–348, (1993).
- [3] K. Smith, 'Neural Networks for Combinatorial Optimization: A Review of More Than a Decade of Research', *INFORMS Journal on Computing*, **11**, No. 1, 15–34, (1999).
- [4] N. Aras, *New Neurocomputational Approaches for Estimating Road Travel Distances and for Solving the Euclidean Traveling Salesman Problem*, Ph. D. Dissertation, Boğaziçi University, 1999.
- [5] N. Aras, İ.K. Altınel, and J. Oommen, 'Kohonen Network Incorporating Explicit Statistics and its Application to the Traveling Salesman Problem', *Neural Networks*, **12**, 1273–1284, (1999).
- [6] İ. K. Altınel, N. Aras, and J. Oommen, 'Fast, Efficient and Accurate Solutions to the Hamiltonian Path Problem Using Neural Approaches', *Computers & Operations Research*, **27**, 461–494, (2000).
- [7] G. Reinelt, *The Travelling Salesman. Computational Solutions for TSP Applications*, Springer-Verlag, Berlin, 1994.
- [8] G. Reinelt, 'TSPLIB—A Traveling Salesman Problem Library', *ORSA Journal on Computing*, **3**, No. 4, 376–384, (1991).