# Graph Theoretic Techniques for Pruning Data and Their Applications

Tetsuya Hoya
Signal Processing and Digital Systems Section,
Dept. of Electrical and Electronic Engineering,
Imperial College of Science, Technology and Medicine,
University of London, SW7 2BT, U.K.
email: t.hoya@ic.ac.uk

**Abstract**

In pattern recognition tasks, we usually do not pay much attention to the arbitrarily chosen training set of a pattern classifier beforehand. This paper proposes several methods for pruning data sets based on graph theory in order to alleviate the redundancy in the original data set whilst retaining the original data structure as far as possible. The proposed methods are applied to the training sets for pattern recognition by a Multi-Layered Perceptron Neural Network(MLP-NN) and the locations of centroids of a Radial Basis Function Neural Network (RBF-NN). The advantage of the proposed graph theoretic methods is that they do not require any calculation for statistical distributions of the respective clusters. The experimental results in comparison with both $k$-means clustering and Learning Vector Quantisation(LVQ) methods show that the proposed methods give encouraging performances in terms of computation for data classification tasks.

Accepted for

**IEEE Transactions on Signal Processing**

*Index Terms:—*

**Data-pruning algorithms, graph theoretic techniques, pattern classification**

[1]Please address correspondence to: Tetsuya Hoya, Laboratory for Advanced Brain Signal Processing, BSI-RIKEN, 2-1 Hirosawa, Wakoh-City, Saitama 351-0198, Japan, electronic mail address: hoya@bsp.brain.riken.go.jp

# 1 Introduction

Neural Networks have played a significant role in pattern classification studies with the rapid development in new computer architectures as well as with advances in vector quantisation networks, Bayesian, or nearest neighbour-based classifiers. In pattern recognition tasks using NNs, normally a large amount of data for training NNs are demanded to generalise the mapping of the input-output relation. This is time consuming. The need for pruning the training data sets is thus desirable, for the sets usually are not chosen with much care and there is therefore a large possibility of containing redundant information in the training phase. Moody and Darken [1] used the centroids of RBF-NNs obtained by $k$-means clustering algorithm. In an earlier work, $k$-nearest neighbour($k$-NN)-based algorithm [2] was used for the reduction in the size of RBF-NNs [3]. This method, however, involves the estimation of the *a posteriori* distributions, which is one of common problems in statistical analysis. Graph theory provides a number of ways to solve problems in a variety of disciplines and has also been used for clustering [4] - [6]. In a recent paper [7], a method to the data reduction was proposed, which automatically selects the number of exemplars for each class. In that paper, an MST/SST(Minimal/Shortest Spanning Tree) was used to calculate the density of the patterns in the training data set of $k$-NN and Learning Vector Quantisation Neural Network(LVQ-NN) classifiers. In this paper, three graph theoretic techniques for data-pruning are proposed and applied to data sets for digit word recognition. It is also proposed that the representative vectors obtained should be suitably chosen to contain "well-balanced" information of the given training data in the present paper. The proposed methods are designed to consider both the hierarchical and non-hierarchical aspects of data sets, and do not require any arduous statistical approximation. In Sections 3 - 5, these pruning methods are developed. The performances of the proposed methods in terms of recognition accuracy are demonstrated where used are the two types of NNs, namely, an MLP-NN and an RBF-NN, each of which is well-known and has been successfully employed in pattern recognition tasks. In the experiment, the recognition tasks are performed with different sizes of the training/centroid vectors. The performances are compared with those obtained by the well-known MacQueen's $k$-means clustering algorithm [8] and LVQ algorithm [9], which can be seen as a natural extension of $k$-means clustering algorithm.

## 2 Formation of the original graph

The graph theoretic data-pruning methods proposed in this paper follows the formation of the pattern vectors into a non-directed tournament graph given below: Provided that we have a pattern set containing $N$ distinct pattern vectors, a non-directed tournament graph $G = (X, A)$ is formed by mapping all the patterns in the pattern set onto vertices of the graph, where $X$ denotes a set of vertices, $x_1, x_2, ..., x_N$, and $A$ is a collection of links $l_1, l_2, ..., l_M$, each of which connects a pair of vertices in $G$. The respective link costs $c_{ij}$ are simply given as the Euclidean distances between the corresponding pairs of vertices, $\boldsymbol{x_i}$ and $\boldsymbol{x_j}$:

$$c_{ij} = \sum_{k=1}^{L} \|x_{ik} - x_{jk}\|_2^2, \ i \neq j, \ \boldsymbol{x_i}, \boldsymbol{x_j} \in X, \tag{1}$$

where $L$ is the length of the pattern vectors. An example of a non-directed tournament graph where $N = 6$ vertices in the graph is illustrated in Fig. 1. It is plausible to think that the pattern space is limited to the original graph obtained by the training set and therefore that the testing patterns should lie somewhere on the original graph. The three graph theoretic pruning methods to be presented in the subsequent sections differ from each other in their ways of partitioning of the original graph into a certain number of its disjoint sub-graphs. The proposed methods then find the absolute centre on the respective sub-graphs so obtained, the location of which indicates the minimum distance from any vertex in the sub-graph. In this sense the absolute centre retains a global information of the entire sub-graph. The absolute centre is found by means of half-way property [10] in the present paper. The choice of the number of sub-graphs is also a significant factor in that the radius of the sub-graph will be decreased as the number of sub-graphs is increased, i.e., the "covering" of the unknown testing patterns by the absolute centre on each sub-graph would be strengthened. This is considered to affect largely the recognition performances by the pattern classifiers.

## 3 Vertex-Chain Method

The data-pruning method, Vertex-Chain method is based upon the graph-partitioning principle in which the original graph is partitioned into some pairs of sub-graphs so that the distances between a dominant vertex and the remaining vertices in the original graph are evenly distributed. This operation is followed by creating a pair of vertex-chains. The pro-

cess of creating a pair of vertex-chains is given as follows:

Step 1: Find the dominant vertex given as $i$-th, $(i = 1, 2, ..., p)$, median vertex of the original graph $G$, and set the vertex as the "seed" vertex of the vertex-chain, $VC_{2i-1}$.

Comment: The $i$-th dominant vertex is chosen since it is considered to have the information in the global sense of the hierarchical data structure.

Step 2: Sort the distances between the dominant vertex and all the remaining vertices in $G$ in an ascending order.

Step 3: Set the vertex on the top of the distance list as the seed vertex of the vertex-chain, $VC_{2i}$.

Comment: The seed vertex is thus chosen as the partitioning point of the original graph. In other words, the risk of losing information by means of graph-partitioning remains to be minimum.

Step 4: For $j = 2$ to $p - 1$, do:

1: If $j$ is even, connect the $j$-th vertex in the list with the tail of $VC_{2i-1}$.

2: Otherwise, connect it with the tail of $VC_{2i}$.

Comment: In this way, the pattern data are distributed evenly on the respective vertex-chains.

In Step 1 above, $i$-th median vertex is such a vertex that gives $i$-th minimum sum of distances between the median vertex and the remaining vertices in the original graph. The dominant vertex may be alternatively given as the absolute centre of the original graph or absolute median(not restricted to a vertex but any point on links), in this paper, however, the $i$-th median vertex is employed for the computational facility. The process of creating the pair of vertex-chains, $VC_1$, $VC_2$, is illustrated in Fig. 2, where 8 vertices in the original graph $G$. In the figure, the respective vertices in $G$, $x_{v_{ij}}$, $(i = 1, 2,\ j = 1, 2, ...4)$, are chained to either vertex-chain $VC_1$ or $VC_2$. The numbers in the respective circles indicate the order of the connection in the process. Vertex-Chain method is then summarised as follows:

Step 1: Form the original graph by the procedure described in the previous section.

Step 2: For $i = 1$ to $p$, do:

1: Create a pair of vertex-chains, $VC_{2i-1}$ and $VC_{2i}$.

2: Partition the original graph into two disjoint sub-graphs, $SG_{2i-1}$ and $SG_{2i}$. The vertices of the sub-graphs, $SG_{2i-1}$, and $SG_{2i}$, are those in the vertex-chain, $VC_{2i-1}$ and $VC_{2i}$, respectively.

3: Find the respective absolute centres of $SG_{2i-1}$ and $SG_{2i}$.

4: Convert the two centres so obtained into the respective representative pattern vectors by the procedure given in Section 6.

In the course of Vertex-Chain method, a combination of $2p$ "well-balanced" data sets is composed, where $p$ is the number of partitioning iterations, and therefore a total of $2p$ representative vectors are respectively obtained in terms of absolute centre. One may intuitively think that as the number of vertex-chains grows the interpolation performance of data points will be improved. This is explained as follows: Provided that a pattern $p_l$, say, is covered by the absolute centre of the sub-graph, $SG_k$ with the distance $d(p_l)$. However, if we have an adequate number of vertex-chains, it is highly possible that $p_l$ is also covered by the absolute centre of other sub-graph(s) with shorter distance than $d(p_l)$. This idea therefore helps us to take the consideration into the improvement in the recognition performance when we have some testing patterns close to the pattern $p_l$. The computational complexity of this algorithm is then $O(max(LN^2/2, pN^3/2))$, where $L$ is the length of each pattern vector and $N$ is the number of vertices.

# 4 List-Splitting Method

In Vertex-Chain method described in the previous section, the partitioning process into sub-graphs *always* begins from the original graph obtained by the procedure described in Section 2. In contrast to Vertex-Chain method, List-Splitting method pays more attention on the hierarchical aspect of the data set in that the original graph is partitioned *recursively* into its $2^p$ sub-graphs, where $p$ is the number of the iterations. List-Splitting method enables us to alleviate the computational load for the procedure of finding an absolute centre of the respective sub-graphs by the reduction in the number of vertices in each sub-graph, which is less than or equal to that of Vertex-Chain method. In addition, this consideration would be advantageous in terms of the interpolation performance in that the absolute centres so ob-

tained will contain "higher-density" information of the entire sub-graph than those obtained by Vertex-Chain method. The following is the summary of List-Splitting method:

Step 1: Form the original graph by the procedure described in Section 2.

Step 2: Set initially the original graph as a parent graph.

Step 3: Repeat $p$ times in the following:

1: Partition the parent graph(s) into a pair of its children graphs by List-Splitting procedure.

2: Set the two children graphs as the parent graphs for the next iteration.

Step 4: Form a total of $2^p$ sub-graphs from the respective children graphs obtained in the previous step.

Step 5: Find an absolute centre on each sub-graph, and convert the $2^p$ centres into the corresponding representative pattern vectors by the procedure given in Section 6.

In Step 3 above, the partitioning is done by List-Splitting procedure. The List-Splitting procedure is the process of splitting the distance list of the parent graph into its two parts. The distance list is a list sorted in an ascending order of distances between the dominant vertex, i.e., the first median vertex, and the remaining vertices in the parent graph. In the respective parts, the total sum of the order of the distances remains to be as equal as possible. In other words, it is considered that the sub-graphs obtained by those two parts have similarly "well-balanced" information in terms of equality in the sum of the distances. List-Splitting procedure is given as follows:

Step 1: Find the dominant vertex of the parent graph, which is given as the first median vertex.

Step 2: Obtain the distance list by sorting the distances between the dominant vertex and all the remaining vertices in the parent graph in an ascending order.

Step 3: For $i = 1$ to $N/2$ do:

1: Include in one of the parts the vertices at both $i$-th and $(N - i)$-th positions from the top of the list. When $N$ is an odd number, include also the vertex of $(floor(N/2) + 2)$-th in it.

2: Include in the other the vertices at both $(i+1)$-th and $(N-(i+1))$-th positions in the list.

where $N$ is the number of vertices in the parent graph. For List-Splitting method, the computational complexity is given as $O(max(LN^2/2, N^3/(2 \cdot 8^p)))$. In Fig. 3, an iteration of List-Splitting procedure is illustrated, where 8 vertices are in the parent graph. In the figure, the numbers in the respective lists indicate the numerical orders of distances from the dominant vertex.

## 5 SST-Splitting Method

The SST-based techniques have been employed in classification tasks [6], [7]. The proposed method described here is considered to take another view of the hierarchical aspect of the data set by the use of an SST of the original graph. SST-Splitting method has some flavour of Minimax SST segmentation method which was previously used for image segmentation [11], but in contrast to the algorithm for the SST segmentation, the tree is cut at the minimum link in the process. This is explained as follows: Provided that a testing pattern is lying somewhere on the link of the cutting point and that we chose the link as the link having the largest cost in the tree, the covering of the testing pattern by the absolute centre to be obtained would be relatively weak. This may lead to the poor performance in reference to the interpolation problem similar to that of Vertex-Chain method. In the paper, the link having a minimum link cost is therefore selected as a cutting point of the SST. In the proposed method, the SST is recursively cut into its $p(< N)$ sub-trees and the original graph is correspondingly partitioned into its $p$ sub-graphs according to the vertices in the respective sub-trees. The SST algorithm for the data-pruning task is then given as follows:

Step 1: Form the original graph.

Step 2: Find its SST, and set initially the SST as $T_{max}$.

Step 3: Repeat $p$ times in the following:

1: Search $T_{max}$ in the forest with the largest number of vertices.

Comment: The tree with the largest number of vertices is selected for splitting into further in order to avoid the sparsity in the data struc-

ture.

2: Cut $T_{max}$ at the minimum cost link and form its two sub-trees.

Step 4: Partition the original graph into its $p$ sub-graphs, according to the vertices in the sub-trees of the SST obtained in the previous step.

Step 5: For each sub-graph, find an absolute centre.

Step 6: Convert the $p$ absolute centres into $p$ representative pattern vectors by the procedure described in Section 6.

For this method, the computational complexity is given as $O(max(LN^2/2, pN^3/3))$.

# 6 Conversion of the absolute centre

In the previous sections, the absolute centre on each sub-graph needs to be subsequently converted into the corresponding representative vector. This is simply done as follows: Assume that the location of the absolute centre on $link(\boldsymbol{x_1}, \boldsymbol{x_2})$ at offset $p$, where $\boldsymbol{x_1} = (x_{11}, x_{12}, ..., x_{1L})$, $\boldsymbol{x_2} = (x_{21}, x_{22}, ..., x_{2L})$, each element of the centre vector $\boldsymbol{x_p}$ is given as:

$$x_{pi} = (x_{2i} - x_{1i}) \frac{\|\boldsymbol{x_1} - \boldsymbol{x_p}\|_2^2}{\|\boldsymbol{x_1} - \boldsymbol{x_2}\|_2^2} + x_{1i}, \tag{2}$$

or, alternatively,

$$x_{pi} = (x_{1i} - x_{2i}) \frac{\|\boldsymbol{x_2} - \boldsymbol{x_p}\|_2^2}{\|\boldsymbol{x_1} - \boldsymbol{x_2}\|_2^2} + x_{2i}. \tag{3}$$

# 7 Experiment

## 7.1 Formation strategy

The graph-theoretic algorithms for data set reduction described in Sections 3 - 5 are applied here. In the experiment, the corresponding methods are applied to prune the training set for Japanese and for English digit word recognition, respectively. For these tasks, there are a couple of possible paradigms to formulate the data-pruning as follows:

Paradigm 1: A non-directed tournament graph $G$ containing vertices, $x_1, x_2, ..., x_N$, is formed, which consists of all the pattern vectors *regardless* of the categories to which the respective vectors belong. A total of $p$ representative vectors are *directly* found on the tournament graph.

Paradigm 2: Where both the number of categories, $\xi$, and the number of the patterns, $\phi(i)$, $(i = 1, 2, ..., \xi)$, which belong to the respective categories are available, $\xi$ distinct sub-graphs $SG_1, SG_2, ..., SG_\xi$, are formed, where $SG_i$ contains $x_{i1}, x_{i2}, ..., x_{\phi(i)}$. The $p$ representative vectors are found *one by one* on each sub-graph, $SG_i$.

In Paradigm 1 above one can intuitively assume that the $p$ representative vectors should at least cover all the categories($p \geq \xi$). In digit word recognition tasks, the number of categories is already known($\xi = 10$) and so is normally the number of patterns in each category. Hence this paradigm is not suitable in terms of computational complexity. Paradigm 2 is, therefore, considered to be suitable for pruning the data set of digit word recognition tasks and therefore is chosen in this paper.

### 7.2 Data sets for digit word recognition tasks

For Japanese digit word recognition, the respective digits in Japanese, /ZERO/, /ICHI/, /NI/, /SAN/, /YON/, /GO/, /ROKU/, /NANA/, /HACHI/, and /KYU/, correspond to /ZERO/, /ONE/, /TWO/, /THREE/, /FOUR/, /FIVE/, /SIX/, /SEVEN/, /EIGHT/, and /NINE/ in English. The speech data used for a training set for Japanese digit word recognition consists of 3 utterances of 5 independent non-trained speakers and were recorded in an ordinary room, and for English the data consisting of 7 utterances of 5 independent speakers were collected from SFS(Speech Filing System) database [12]. The numbers $\phi(i)$ in Paradigm 2 are therefore fixed as $\phi(i) = 3, i = 1, 2, ..., 10$ for Japanese and $\phi(i) = 7$ for English recognition, respectively. For testing, a total of 150 data were collected from the same speakers and used for both Japanese and English recognition tasks. The input vector of neural networks is the encoded speech data by LPC-mel-cepstral analysis with the pre-process using a seven-order adaptive inverse filter developed by Nakajima *et al* [13] and normalised within the range from -1.0 to 1.0 before training.

### 7.3 Classifiers

As classifiers, an MLP-NN and an RBF-NN were used for the recognition tasks. A three-layered neural network with 256 units in the input, 30 in the hidden, and 10 units in the output layer, each of which corresponds to the threshold units of the respective digits, was used [14]

as an MLP-NN classifier and trained by the well-known back-propagation algorithm [15] with adjustable momentum-term method [16], [17]. For an RBF-NN classifier, a Generalised Regression Neural Network(GRNN) [18] with multiple output units was used, which is based on a well-established nonlinear regression theory, and in this scheme there is no requirement of the iterative training procedure: the weight vectors between hidden and output layers are just fixed as target vectors [19]. The number of units in the input and output layers were fixed as those for the three-layered neural network, and the number of centroids was varied to evaluate the recognition performance. The centroid vectors are found among all patterns in the training sets.

### 7.4 Experimental results

The performances for Japanese digit word recognition are tabulated in Table 3 with an MLP-NN and in Table 4 with an RBF-NN, respectively, and the comparison of the computation time is shown in Fig. 5. For English digit word recognition, the performances are in Table 1 with an MLP-NN and in Table 2 with an RBF-NN, respectively, and the computation time is in Fig. 4. The values of the standard deviation for the RBF-NN used are also tabulated as indicated by $\sigma$ in the respective tables. The corresponding values for the respective numbers of RBFs were chosen, which gives the best performances with 20 RBFs by the respective pruning methods within the range from $\sigma = 0.0$ to $\sigma = 10.0$. (Note that the numbers of representative patterns/RBFs selected for English digit word recognition are different from the numbers for Japanese since the size of the original training sets are different, i.e., the total number of patterns in the original set for the English is 150, whilst the number is 350 for Japanese.) The recognition results by MacQueen's $k$-means clustering algorithm and the case where a complete training set is used by an MLP-NN are also tabulated in the respective tables, for comparison. The recognition performances obtained by the proposed graph-theoretic methods are satisfactory in both Japanese and English digit word tasks. In Table 1 and Table 2, the performances in English digit word recognition task using SST-Splitting method are particularly encouraging in comparison with both $k$-means clustering and LVQ algorithms. The experiments of LVQ algorithm were performed using LVQ_PAK [20]. The initial codebook was obtained using *propinit* LVQ_PAK algorithm, then the respective representative vectors of LVQ algorithms were obtained using LVQ1 after 1000 training steps with the learning rate

0.1.

## 8 Conclusion

In this paper, three graph theoretic data-pruning methods have been described with the applications to the training data for Japanese and English digit word recognition tasks using an RBF-NN and an MLP-NN. The advantages of these algorithms are that there is no need for the calculation for the *a priori* statistical distribution of each cluster whilst retaining satisfactory recognition performances. However, as shown in the figures, due to the nature of the procedure of finding absolute centre by the use of half-way property relatively a large number of computational steps for the proposed methods are required for relatively small numbers of representative vectors. In addition, Vertex-Chain method may suffer from the expensive computation as the number of vertices in the original graph grows, since the absolute centres are found on each sub-graph in which has $N/2$ vertices, whilst the number of vertices in List-Splitting or SST-Splitting method can be less than $N/2$. Future work will be directed towards the investigation of the efficient methods for finding such centres.

## Acknowledgement

The author would like to thank the reviewers for their insightful and useful suggestions/comments. The author also would like to thank Prof. A. G. Constantinides, Imperial College of Science, Technology, and Medicine, University of London, for his eminent directions and encouragement toward this work which was done as a part of the author's work towards a Ph.D degree.

## References


[1] J. Moody and C. J. Darken, "Fast learning in networks of locally-tuned processing units", Neural Networks, Vol.6(4), pp.525-533, 1989.

[2] P. A. Devijer and J. Kittler, "Pattern Recognition: A Statistical Approach", Prentice Hall International, 1982.

[3] M. A. Kraaijveld and R. P. W. Duin, "Generalization Capabilities of Minimal Kernel-Based Networks", Proc. of Int. Joint Conf. on Neural Networks , Seattle, 1991.

[4] J. G. Augustson and J. Minker, "An Analysis of Some Graph Theoretical Cluster Techniques", Jl. of ACM, vol. 17, No. 4, pp.571-588, Oct. 1970.

[5] D. W. Matula, "Classification and Clustering: Graph theoretic techniques for cluster analysis algorithms", in J. Van Ryzin, Ed. New York: Academic Press, pp. 95-129, 1977.

[6] C. T. Zahn, "Graph-theoretic methods for detecting and describing gestalt clusters", IEEE Trans., C-20, pp. 68-86, 1971.

[7] H. Tahani, B. Plummer, and N. S. Hemamalini, "A New Data Reduction Algorithm for Pattern Classification", ICASSP'96, pp. 3446-3449, 1996.

[8] J. B. MacQueen, "Some methods for classification and analysis of multivariate observations", Proc. Symp. Math. Statist. and Probability, 5th, Berkeley, Vol. 1, pp.281-297, AD 669871, Univ. of California Press, Berkeley, 1967.

[9] T. Kohonen, "Improved versions of learning vector quantization", Proc. of Int. Joint Conf. on Neural Networks, Vol. 1, pp.545-550, San Diego, CA, 1990.

[10] N. Christofides, "Graph Theory: An Algorithmic Approach", Academic Press, 1975.

[11] O. J. Morris, M. de J. Lee, and A. G. Constantinides, "Graph theory for image analysis: an approach based on the shortest spanning tree", IEE Proc., Vol. 133, Pt. F, No. 2, Apr. 1986.

[12] M. Huckvale, "Speech Filing System Vs3.0 – Computer Tools For Speech Research", University College London, Mar. 1996.

[13] T. Nakajima, T. Suzuki, H. Ohmura, S. Ishizaki, and K. Tanaka, "Estimation of vocal tract area function by adaptive deconvolution and adaptive speech analysis system", The Jl. of the Acoustical Soc. of Japan,

[14] T. Hoya, H. Ihara, T. Asano, and Y. Ishida, "Word Recognition by Neural Networks Using Adaptive Mel-Cepstral Analysis", Proc. of Int. Workshop on Intelligent Signal Proc. and Comm. Syst., pp.68-71, Seoul, 1994.

[15] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation." In Parallel Distributed Processing: Explorations in the Microstructure of Cognition ( D. E. Rumelhart and J. L. McClelland, eds.), Vol. 1, Chapter 8, Cambridge, MA: MIT Press, 1986. Vol. 34, No. 3, pp. 157, 1978(in Japanese).

[16] S. Haykin, "Neural Networks: A Comprehensive Foundation", Macmillan College Publishing Co. Inc., 1994.

[17] K. Nakano, K. Inuma, and Neuron-net group, "Introduction and practice: Neuro Computer", Gijutsu Hyoron Press, 1989(in Japanese).

[18] D. F. Specht, "A general regression neural network", IEEE Trans. on Neural Networks, Vol. 2(6), pp.568-576, 1991.

[19] P. D. Wasserman, "Advanced Methods in Neural Computing – in Chapter 8, Radial Basis-Function Networks", pp.147-176, Van Nostrand Reinhold, New York, 1993.

[20] T. Kohonen, J. Hynninen, J. Kangas, J. Laaksonen, and K. Torkkola, "LVQ_PAK: the learning vector quantisation program package – Version 3.1", Apr. 1995.

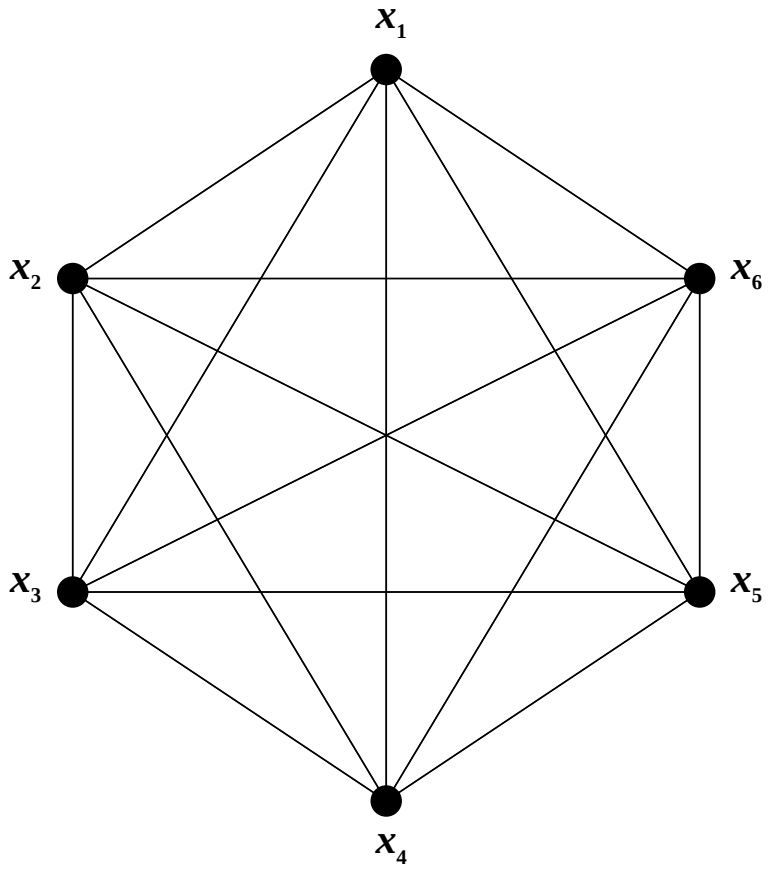


Figure 1: A non-directed tournament graph of the case $N = 6$

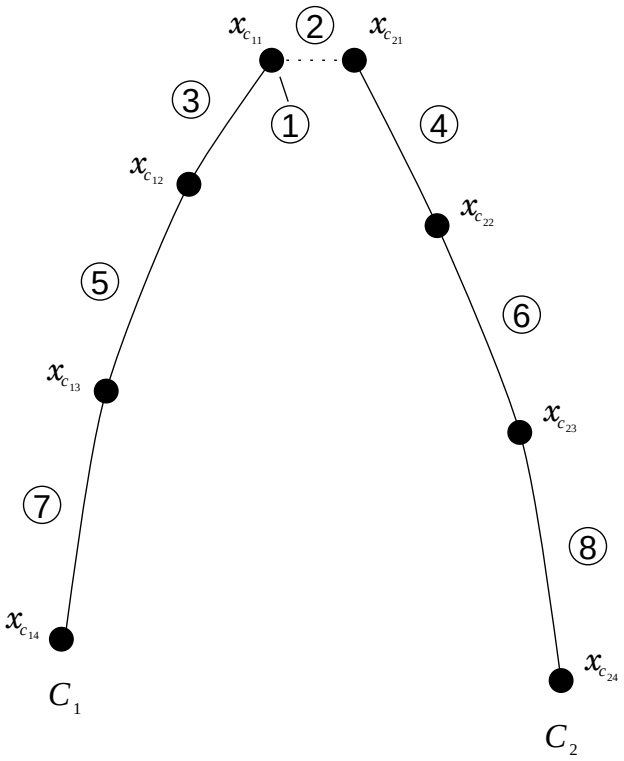


Figure 2: Illustration of an iteration of the connection process: 8 vertices in $G$

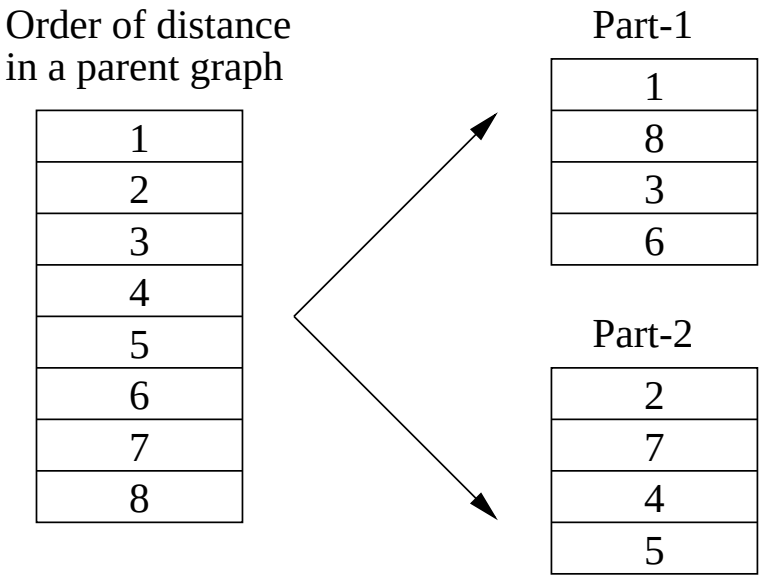


Figure 3: Illustration of a list-splitting process: 8 vertices in a parent-graph

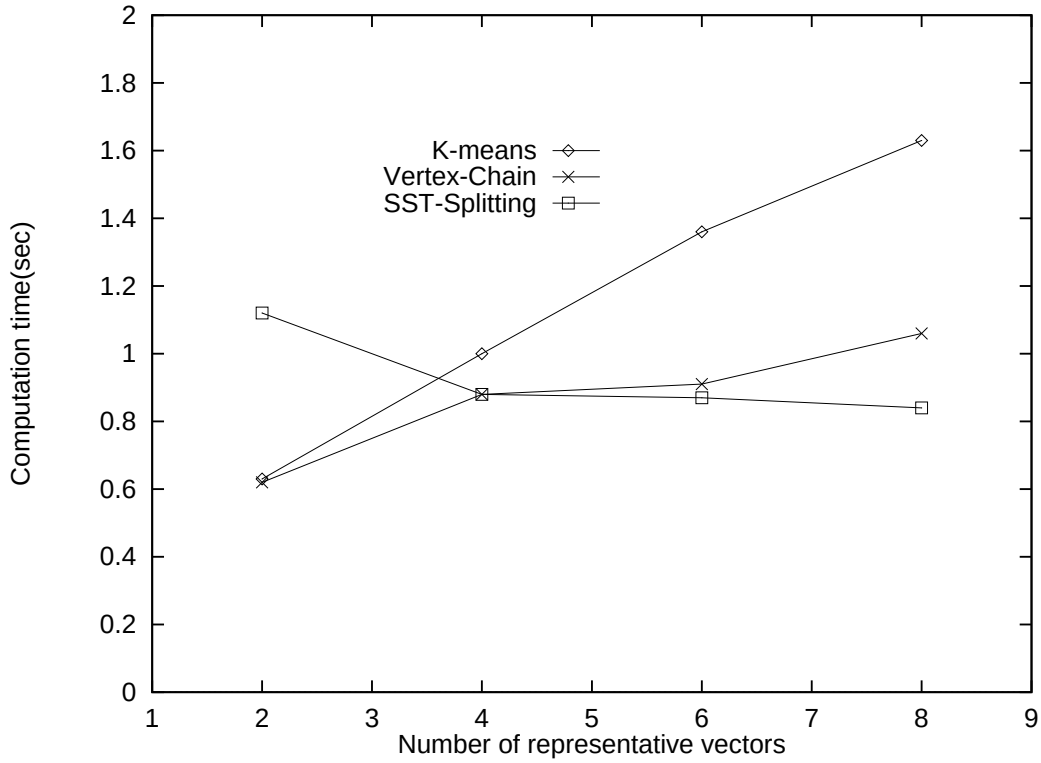


Figure 4: Computation time for pruning the training set for Japanese digit word recognition

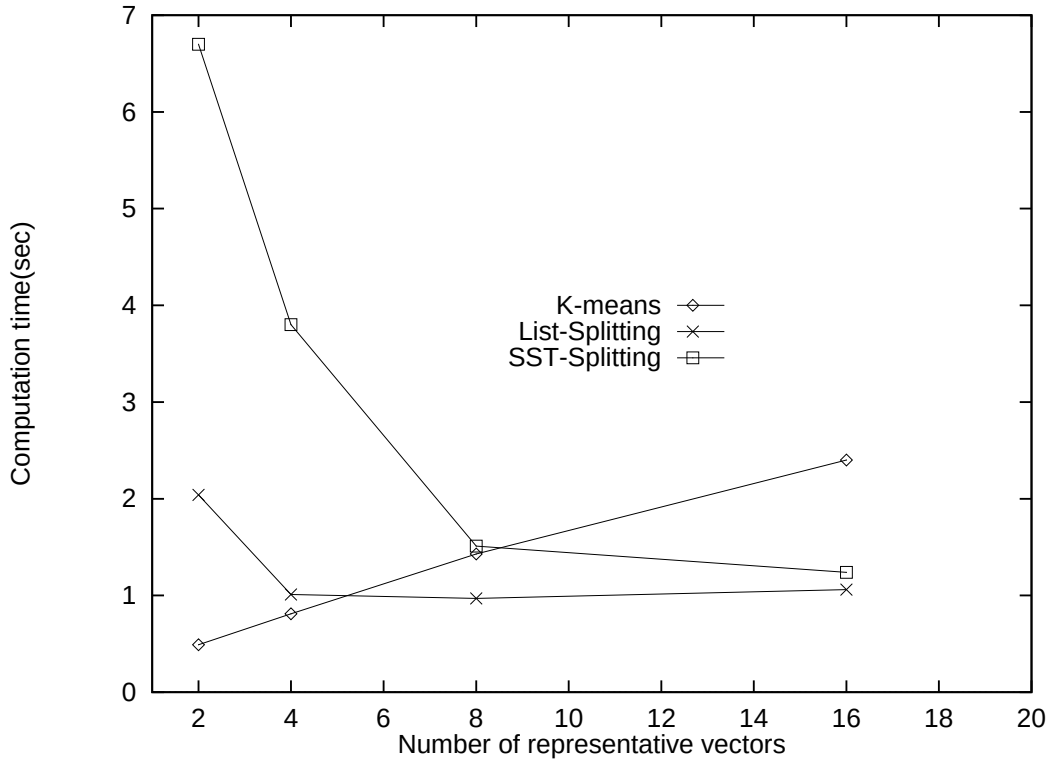


Figure 5: Computation time for pruning the training set for English digit word recognition

| Method | Errors(Using Full Set: 5 errors) | | | |
|---|---|---|---|---|
| | 2 vectors | 4 vectors | 8 vectors | 16 vectors |
| $k$-means | 47/150 | 33/150 | 18/150 | 11/150 |
| LVQ Algorithm | 32/150 | 24/150 | 17/150 | 12/150 |
| Vertex-Chain | 35/150 | 26/150 | 27/150 | 11/150 |
| List-Splitting | 33/150 | 16/150 | 11/150 | 7/150 |
| SST-Splitting | 31/150 | 23/150 | 6/150 | 3/150 |

Table 1: Recognition performance for English digit word recognition with an MLP-NN

| Method | Errors | | | | $\sigma$ |
|---|---|---|---|---|---|
| | 20 RBFs | 40 RBFs | 80 RBFs | 160 RBFs | |
| $k$-means | 47/150 | 44/150 | 30/150 | 12/150 | 1.0 |
| LVQ Algorithm | 30/150 | 28/150 | 20/150 | 13/150 | 1.4 |
| Vertex-Chain | 33/150 | 26/150 | 26/150 | 21/150 | 0.5 |
| List-Splitting | 32/150 | 22/150 | 17/150 | 9/150 | 1.8 |
| SST-Splitting | 36/150 | 16/150 | 9/150 | 9/150 | 2.0 |

Table 2: Recognition performance for English digit word recognition with an RBF-NN

| Method | Errors(Using Full Set: 1 error) | | | |
|---|---|---|---|---|
| | 2 vectors | 4 vectors | 6 vectors | 8 vectors |
| $k$-means | 3/150 | 3/150 | 0/150 | 1/150 |
| LVQ Algorithm | 3/150 | 2/150 | 1/150 | 1/150 |
| Vertex-Chain | 10/150 | 3/150 | 4/150 | 2/150 |
| List-Splitting | 7/150 | 3/150 | N/A | 2/150 |
| SST-Splitting | 5/150 | 1/150 | 1/150 | 1/150 |

Table 3: Recognition performance for Japanese digit word recognition with an MLP-NN

| Method | Errors | | | | $\sigma$ |
|---|---|---|---|---|---|
| | 20 RBFs | 40 RBFs | 60 RBFs | 80 RBFs | |
| $k$-means | 4/150 | 4/150 | 3/150 | 3/150 | 5.0 |
| LVQ Algorithm | 5/150 | 6/150 | 3/150 | 3/150 | 6.0 |
| Vertex-Chain | 8/150 | 4/150 | 5/150 | 4/150 | 2.0 |
| List-Splitting | 4/150 | 5/150 | N/A | 2/150 | 20.0 |
| SST-Splitting | 6/150 | 4/150 | 3/150 | 3/150 | 3.0 |

Table 4: Recognition performance for Japanese digit word recognition with an RBF-NN