# Automated Network Drawing Using Self-Organizing Map

Xiangjun Xu Mladen Kezunovic*
Electrical Engineering Department, Texas A&M University
College Station, TX 77843-3128, USA
**(Phone) 979-845-7509, (Fax) 979-845-9887, kezunov@ee.tamu.edu**

ABSTRACT: In this paper, a method for automatically creating circuit schematic diagrams from the topological information contained in network data files has been proposed. This method is based on Self-Organizing Map (SOM) neural network and the basic idea behind the method is to let the network span itself according to a given "shape" of the network grid. The topology of a network is defined by the connections between its nodes. By forming an SOM using the network connection topology and training it using data grids of desired "shape", the positions of the nodes and their neighbors will be gradually updated until a desired diagram has been created.

Keywords: Visualization, Graph Drawing, Layout, Routing, Self-Organizing Map, Artificial Neural Network.

## I. INTRODUCTION

Automated graph drawing has been studied many years and become relatively mature, especially for directed and layered graphs. R. Tamassa addressed the problem of readability of automatically created diagrams in [1] and a procedure for automatically drawing directed graphs was presented in [2], which was based on the clan-based graph decomposition. In [3], the author proposed an automatic method for drawing compound digraphs that contain both inclusion and adjacency edges. There are also some graph drawing tools available. In [4], an X window based visualization tool for automatic generation of high-quality drawings of directed graphs has been introduced. AT&T Bell Laboratory [5] and University of Saarlandes [6] also have automated graph drawing tools. Automated routing plays a important role in VLSI layout [7]. Due to the hierarchical relationships existing in the directed graphs, they are not very suitable for power system network visualization, where all the nodes are at the same level.

As a contrast, fewer studies were done about automated graph drawing in the power system realm. The users may have many old power network data files without graphical representation information. It is meaningful to find a way to automatically create graphical representations of those data files. For example, ATPDraw [8] provides a graphics user interface to create and edit circuit diagrams. Those ATPDraw circuit files (.CIR) contain both the layout information and the component parameters. ATPDraw can convert those circuit files into ATP files (.ATP) before calling the ATP [9] program. On the other hand, the ATP files have no layout information. So, the process can not be reversed, i.e. the function to recover an ATPDraw circuit file from an ATP file is not available. Since the graphical representation of an electric circuit has many benefits when compared to just the data file, it is meaningful to find a method, which can automatically create a graphical representation for a give data file. Such kind of function of ATPDraw has been demanded for a long time.

Guise [10] developed a program that can automatically create schematic diagrams from EMTP data files. Since the problems are "NP-complete", heuristics are used to create the drawing initially, and then the user can adjust the drawing manually. In [11], Kolysh proposed a method to create substation scheme graphs from substation data files automatically. The method utilized the fact that there are only a few well-known types of substation configurations, which is not applicable in general networks.

In this paper, a method, which uses Self-Organizing Map (SOM) neural network [12] to generate the graphical representation of the network data files, will be proposed. The basic idea behind the method is to let the network span itself according to a given shape of the network grid. The paper is organized as follows: Automatic graph drawing is briefly introduced at first, followed by the introduction of SOM and the proposed method. Also an example of using this approach in creating the network diagrams from IEEE 30-bus system is given. At last, the selection of training parameters, data grid "shape", data presentation sequence and stop criteria has been discussed.

## II. AUTOMATIC GRAPH DRAWING

A graph can be defined by $G(V,E)$, where $V$ is a set of vertices (or nodes) and $E$ a set of edges (or connections). The central problem of Automated Graph Drawing is to create a drawing which looks as pleasing as possible from a given set of nodes and their connections. Some commonly used aesthetic criteria are: minimum crossing, minimum bends,

maximum of symmetries, optimal usage of layout area, etc. [1].

Graphs with edges of directions are called digraphs. Several automatic graph-drawing algorithms have been proposed for digraphs [2][3]. For general undirected graphs, the most commonly used method is the *spring embedder* algorithm, which may show poor performance for graphs containing dense subgraphs [1].

There are usually two steps to create a drawing from a given set of data. First, the nodes must be arranged at their optimal positions. Second, the connections among those nodes should be routed in an aesthetic manner to improve the visibility.

It is possible to arrange the nodes to avoid line crossing for planar graphs. Also for un-planar graphs, minimum line crossing can also be used as a criteria to layout the nodes, although the problem is "NP-complete" and certain heuristic methods must be used to solve the optimization problem. There may be some other requirements on the node positions. For example, it may be required to locate the nodes on a grid.

Routing the connections is very application oriented. For example, some applications require that the connection lines are straight. In VLSI applications, orthogonal routing is used [7].

For a big graph, it is often desirable to divide it into to smaller subgraphs and process each part separately. One heuristic method for graph partitioning has been given in [13].

Automatic graph drawing is still an open problem. No existing method is suitable for all situations. In this paper, a new algorithm based on Self-Organizing Map, which can be used for automatically creating electric circuit diagram, is proposed. Unlike control systems, circuit diagrams are undirected graphs and the methods given in [2-4] are not applicable.

In this paper, we will focus only on the node layout problem and simply connect the nodes using straight lines after the positions of the nodes have been obtained.

## III. NETWROK DRAWING USING SELF-ORGANIZING MAP

A method based on Self-Organizing Map (SOM) is described in this section. Compared with the existing automatic graph drawing methods [1-3,10,11], the new method is more adaptive. For a given network, its nodes are seen as neurons and the whole network forms a SOM. By presenting training data to the SOM, it will automatically expand to form a circuit diagram.

### A. Self-Organizing Map

Kohonen's Self-Organizing Maps (SOM) are neural networks consisting of a single layer of neurons (or nodes) [12]. The neurons in a SOM are usually organized into a one or two-dimensional structure, which is called the topology of the map. The commonly used two-dimensional topologies are grid and hex. Every neuron in a map receives the same input $\mathbf{x}^k \in R^n$ and has an n-dimensional weight vector $\mathbf{w}_i \in R^n$ associated with it. The competitive learning rule to update the weight is:

$$\Delta\mathbf{w} = \rho\phi(r_i, r_{i*})\,(\mathbf{x}^k - \mathbf{w}_i) \text{ for all } i=1,2, \ldots s$$

where $i^*$ is the index of the winner unit, $\rho$ is the learning rate, and $\phi(r_i, r_{i*})$ is the neighborhood function.

Applying all the training data once to the map is called one epoch. After the training, the weights of the neurons will be updated to resemble the input data and similar neurons will be grouped into neighborhoods. SOMs are often applied in categorizing the data or visualizing the patterns in the data of high dimensions.

Usually, two phases are involved in the training process. In the ordering phase, the neuron weights are expected to order themselves in the input space consistent with the associated neuron positions. Higher learning rate and larger neighborhood are used for this phase. During the tuning phase, the weights are expected to spread out relatively evenly over the input space while retaining their topological order found during the ordering phase.

### B. Proposed Graph Drawing Algorithm

The prime task for automatic circuit diagram drawing of a given electric network is to determine the positions of its nodes. The connections among nodes define the topology structure, which is the most important property of a network. If we use neurons to represent the network nodes and connect the neurons in the same network topology, a SOM will be formed. The weights of neurons will be the coordinate positions of the nodes. Actually, in this specific application, neuron weights, node positions and coordinates are only different views of the same thing.

Initially, all the neurons in the SOM will have either random or the same weights (positions). The weights will be updated using the competitive learning rule. If the training points are equally distributed in the drawing area, the network will expand and adjust its node positions to occupy the whole area. The overall procedure of the algorithms is as follows:

Step 1. Read in the network topology
Step 2. Form SOM according to the network topology
Step 3. Create training data grid, which specifies the drawing area
Step 4. Train the SOM and obtain the coordinates of all nodes

In the following three sections, step 2, 3, and 4 will be discussed in detail.

### C. Network Connection Topology

In a SOM, topology is used to determine the neighbors of a neuron. When a neuron's position is updated, the positions of

its neighbors will also be updated. The commonly used topologies are grid and hex, which have four and six neighbors per neuron respectively as shown in Fig. 1.

The topology of a network is determined by the connections between its nodes. When the coordinates of a node are changed, its neighboring nodes should also be "dragged" along it. Therefore, it is reasonable to define the neighborhood function according to the connections between network nodes. We call the topology of the SOMs that adopts network connections for describing their neighborhood relationship the *connection topology*.

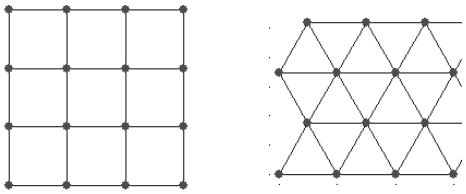

Fig. 1. grid and hex topologies

A connection topology can be represented using a matrix. The elements of the matrix are the minimum connections between two nodes. Its elements $D_{ij}$ are defined by:

$$D_{ij} = \begin{cases} 0 & if\ i == j \\ 1 & if\ i\ and\ j\ are connected directly \\ 2 & if\ D_{ik} = D_{kj} = 1,\ \exists k \\ \vdots & \vdots \\ n & if\ D_{i k1} = D_{k1 k2} = ... = D_{kn j} = 1, \exists k1, k2, ... kn \end{cases}$$

The connection topology of a network can be obtained easily from its binary bus connection matrix **B**.

D. Training Data

The training data used is a grid which spans the whole drawing area as shown in Fig. 2.

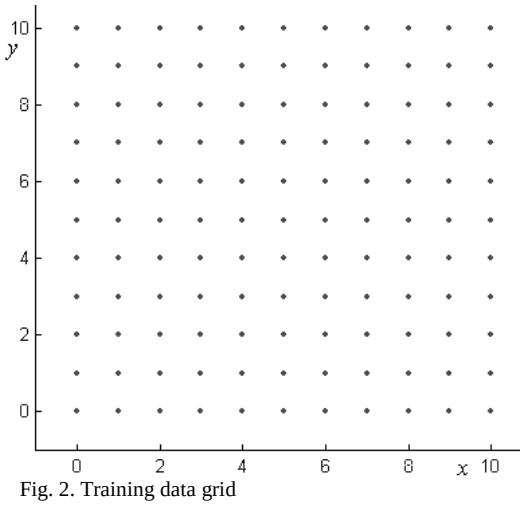


Fig. 2. Training data grid

The network will expand according to the training points in the training data set. One requirement about the number of points is that it should be bigger than the number of nodes. Otherwise, some nodes are not separable. On the other hand, too many points will slow down the training speed. Here, we select the number of training points as three to four times the number of the nodes.

E. Competitive Learning of SOM

The training process of a SOM is depicted in Fig. 3. The training point *p* is presented to the SOM and the neuron with weight closest to the training point is chosen as the winner. Then the weights of the winner and the neighboring neurons are updated. The neighborhood is determined by the connection topology of the network as given in section C. This updating process will be repeated until predefined training steps have been reached or certain criteria has been met.

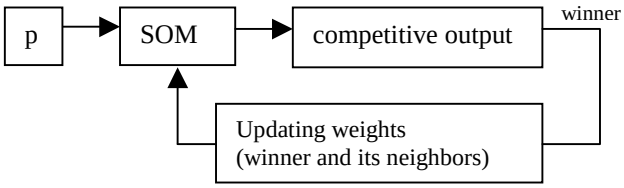


Fig. 3. Competitive learning

There are two training parameters controlling the weight updating process. The *Learning Rate* (LR) gives the updating rate. And the *Neighborhood Distance* (ND) sets the updating range. The training process is often divided into two phases, i.e. the ordering phase and the tuning phase, according to the different training parameters adopted.

F. Application Examples

The SOM-based graph drawing algorithm has been applied to the IEEE 30-bus system. The coordinates of all the nodes are initialized to (0,0). The training data is the grid as shown in Fig 2.

Initially, the node positions are set to zero, so all the nodes overlap as shown in Fig. 4. The result after applying the first two training points (0,0) and (0,1) is given in Fig. 5. Since the point (0,0) is overlapped with the initial node positions, it has no effect on the result. After the point (0,1) being presented, the winner is the first node, node #1. Its position is updated to (0,0.9) and its neighboring nodes are also "dragged" along with it.

Applying the whole training data set once to the map is called one epoch. The result after 700 epochs is given in Fig. 6.

As we can see from Fig. 6, the number of edge crossings is one, which is not optimal as the IEEE 30-bus system is a planar graph and can be drawn with zero edge crossing. The

crossing happens between edge 4-12 and edge 10-17. If we move node #16 below edge 4-12, the optimal crossing number will be achieved. But, considering that there is no explicit objective on minimizing edge crossings in the algorithm, and that the edge crossing number obtained is small (only one crossing), the final result is pretty good.


Fig. 4. Initial node positions of IEEE 30-bus system

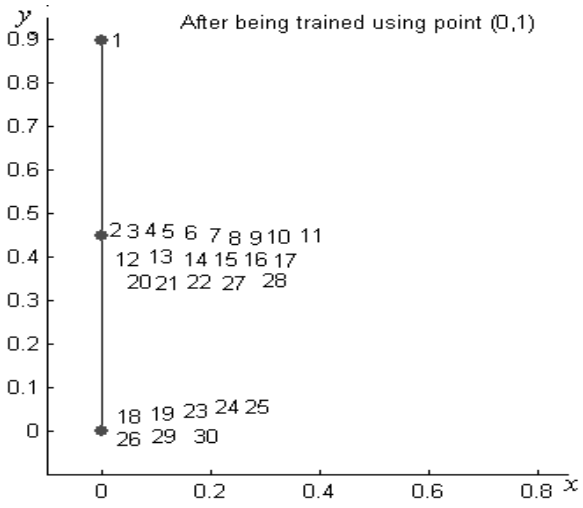


Fig. 5. Node positions of IEEE 30-bus system after two training steps (ordering LR = 0.9, ordering ND=5)

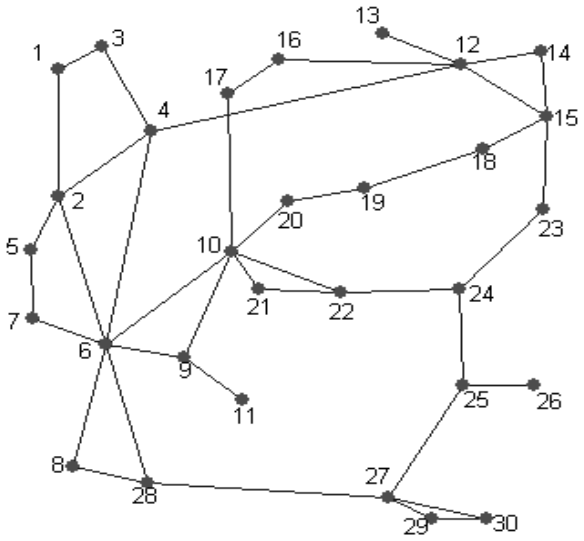


Fig. 6. Automatically created graphical representation of IEEE 30-bus system (ordering epochs = 400, ordering LR = 0.9, ordering ND=5, tuning epochs =300, tuning LR=0.2, tuning ND=1)

## IV. FURTHER DISCUSSION

### A. Training Parameters

The learning rate, which ranges from 0 to 1, will determine the update rate. Larger learning rates will change neuron weights (node positions) more dramatically. That is one of the reasons to generally have two separate phases for SOM learning. In the ordering phase, bigger learning rates are used, while smaller learning rates are used for the tuning phase.

The neighborhood distance determines the range of each update, whose value may range from 0 to the maximum value of the connection matrix. For zero neighborhood distance, only the winner neuron itself will be updated. For neighborhood distance of one, both the neuron and its direct neighboring neurons will be updated.

The learning rate and the neighborhood distance may be adjusted constantly according to certain algorithms. The network shown in Fig. 6 has been trained using the same training algorithm as in the Matlab Neural Network toolbox [14].

Assuming $ND_{ordering}$ and $ND_{tuning}$ are the neighborhood distances of the ordering phase and the tuning phase respectively. $LR_{ordering}$ and $LR_{tuning}$ are the learning rates of the ordering phase and the tuning phase respectively. $EPOCHS_{tuning}$ is the epochs for the ordering phase. The algorithm to adjust the learning rate (LR) and neighborhood distance (ND) of the ordering phase are:

$$ND = 1.00001 + (ND_{ordering} - 1) * (1 - epoch / EPOCHS_{ordering})$$

$$LR = LR_{tuning} + (LR_{ordering} - LR_{tuning}) * (1 - epoch / EPOCHS_{ordering})$$

The learning rate (LR) and neighborhood distance (ND) of the tuning phase are adjusted by:

$$ND = ND_{tuning} + 0.00001$$

$$LR = LR_{tuning} * EPOCHS_{ordering} / epoch)$$

The resulted LR and ND have been given in Fig. 7.

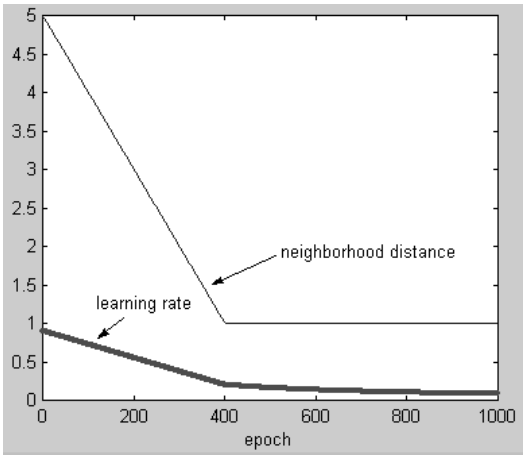


Fig. 7. Learning rate and neighborhood distance

In Fig. 7, the vertical axis is the learning rate and neighborhood distance, and the horizontal axis is the training epoch. The two phases of training can be clearly seen in Fig. 7 from the turning points of the training parameters.

B. Shapes of the Training Data

The neuron weights (i.e. the node positions) will be updated according to the shape of the input training data. When the training is finished, the network will expand to the area covered by the training data.

The shape of the training data can be used to control the desired network shape. For example, knowing that a network consisting of two weakly connected geographical portions, we can use the training data with the desired shape.

Also the training data can be used to represent some constraints on the drawing area. For example, if the right bottom corner needs to be left blank, a training data grid without points at the right bottom corner can be used. An example has been given in Fig. 8 and Fig. 9.

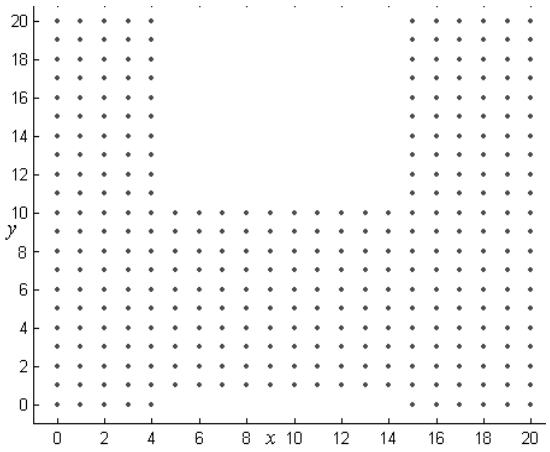


Fig. 8. Training data grid

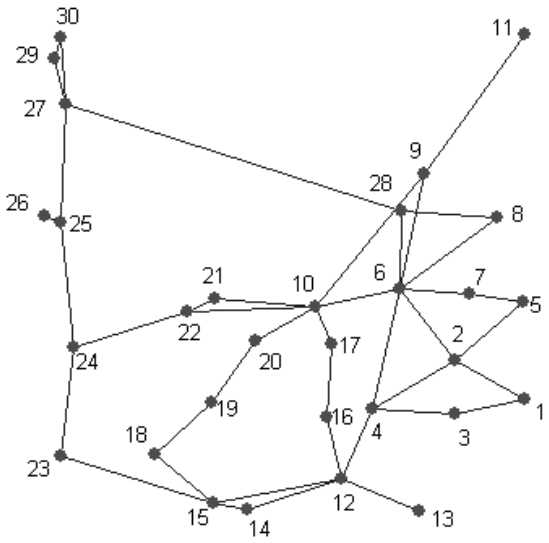

Fig. 9. Automatically created graphical representation of IEEE 30-bus system (ordering epochs = 600, ordering LR = 0.9, ordering ND=5, tuning epochs =700, tuning LR=0.2, tuning ND=1)

The training data grid of the shape shown in Fig. 8 has been used to train the SOM. The result is given in Fig. 9. As we can from Fig. 9, the positions of the nodes have been arranged in a manner to avoid the vacant training area.

C. Data Presentation Sequences

In the process of creating Fig. 6, the training data are presented sequentially. Other presentation sequences are also possible. For example, the Neural Network toolbox uses random sequence to present the data, which will create different diagram every time even the same set of training data is used. Fig. 10 shows a result of presenting the training data randomly.

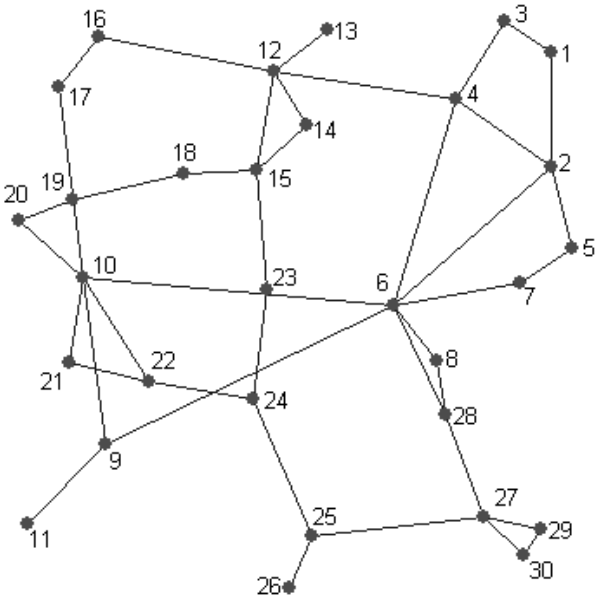

Fig. 10. Automatically created graphical representation of IEEE 30-bus system (traing parameters are same as Fig.6 except the training data are presented randomly)

When the training begins, all the neurons have been initialized to have the same weight, so the winner will be the first neuron. By changing the node ordering, we can control the first node to be chosen. But the ordering does not affect the result once the nodes no longer overlap. Node ordering algorithms such as column minimum degree permutation are not utilized.

D. Criteria for Stopping

The stop criteria is used to determine if the training has been accomplished. A fixed training epoch is often used as the stop criteria. For example, the training in Fig. 6 utilizes 400 epochs for the ordering phase and 300 epochs for the tuning phase.

Certain aesthetic rules can be used as the stop criteria. For

example, the number of edge crossings, the minimum separation distance between nodes, etc.

## V. CONCLUSION

In this paper, a new automatic graph drawing method has been proposed. The new method is based on the competitive learning of Self-Organizing Map. By updating the node positions continuously using the training points, the network will expand according to the shape of the training data grid.

The new method works well for undirected graphs, such as electric circuit diagrams, which are usually more difficult for automatic graph drawing than directed graphs.

The proposed method can easily handle constraints. For example, drawing area constraints can be easily achieved by using the training data grid of desired shape.

Combining with the existing methods of graph partitioning, the proposed method may be able to handle very large graphs.

## VI. REFERENCES


[1] R. Tamassia, G. D. Battista, and C. Batini, "Automatic graph drawing and readability of diagrams," *IEEE Trans. on Systems, Man, and Cybernetics*, vol. 18, Jan./Feb. 1988.

[2] C. L. McCreary, R. O. Chapman, F.S. Shieh, "Using graph parsing for automatic graph drawing," *IEEE Trans. on Systems, Man, and Cybernetics – Part A : Systems and Humans*, vol. 28, Sep. 1998

[3] K. Sugiyama, K. Misue, "Visualization of structural information: automatic drawing of compound digraphs," *IEEE Trans. on Systems, Man, and Cybernetics*, vol. 21, July/August 1991.

[4] daVinci, http://www.tzi.uni-bremen.de/~davinci/

[5] GraphViz, http://www.research.att.com/sw/tools/graphviz/

[6] VCG tool, http://www.cs.uni-sb.de/RW/users/sander/html/gsvcg1.html

[7] E. S. Kuh, T. Ohtsuki, "Recent advances in VLSI layout", Proceedings of the IEEE, vol. 78, Feb. 1990.

[8] ATPDraw user manual, avalaible at ATP secure ftp site

[9] ATP website, http://www.emtp.org/

[10] N. De Guise, G.Paris, M.Rochefort, "Extending a real-time power system simulator's graphical user interface," ICDS'97, Second International Conference on Digital Power System Simulators, Montreal, Quebec, Canada, May 28-30, 1997

[11] A. L. Kolysh, M. A. Slonim, "Automatic process for substation scheme generation based on ASCII load flow files,"

[12] Teuvo Kohonen, Self-Organizing Maps, 3rd ed., Springer, 2001. Berlin, New York

[13] B. W. Kernighan and S. Lin , "An efficient heuristic procedure for partitioning graphs", [online] http://netlib.bell-labs.com/cm/cs/who/bwk/partitioning/index.html

[14] Neural Network Toolbox User's Guide, Mathworks, 1998

[15] Peter Eades, Roberto Tamassia, "Algorithms for drawing graphs: an annotated bibliography", technical report CS-89-09, Brown University, Oct. 1989.


## VII. BIOGRAPHIES

Xiangjun Xu (S'99) received his B.E and M.E. degrees from Southeast University and Shanghai Jiaotong University, all in electrical engineering, in 1992 and 1995 respectively. After that, he worked as a teacher in Shanghai Jiaotong University. Since Sep. 1998, he has been with Texas A&M University pursuing his Ph.D. degree. His research interests are computer application on power systems, signal processing, artificial intelligence, and event analysis.

Mladen Kezunovic (S'77, M'80, SM'85, F'99) received his Dipl. Ing. Degree from the University of Sarajevo, the M.S. and Ph.D. degrees from the University of Kansas, all in electrical engineering, in 1974, 1977 and 1980, respectively. Dr. Kezunovic's industrial experience is with Westinghouse Electric Corporation in the USA, and the Energoinvest Company in Sarajevo. He also worked at the University of Sarajevo. He was a Visiting Associate Professor at Washington State University in 1986-1987. He has been with Texas A&M University since 1987 where he is the Eugene E. Webb Professor and Director of Electric Power and Power Electronics Institute. His main research interests are digital simulators and simulation methods for equipment evaluation and testing as well as application of intelligent methods to control, protection and power quality monitoring. Dr. Kezunovic is a registered professional engineer in Texas, and a Fellow of IEEE.