A Novel Method for Computing the Minimum Spanning Tree and Solution Based on Cellular Neural Networks Implemented on Digital Platforms

Do Trong Tuan[1]
1-Faculty of Electronics and Telecommunications
Hanoi University of Technology
Hanoi - Vietnam
e-mail: dotuan@mail.hut.edu.vn

J. C. Chedjou[2] , Alireza. Fasih[2], K. Kyamakya[2]
2-Institute for Smart-Systems Technologies
University of Klagenfurt
Klagenfurt - Austria
e-mail:{jean.chedjou, alireza.fasih,
kyandoghere.kyamakya}@uni-klu.ac.at

***Abstract*— We present a novel method for transforming graph theoretical problems into optimization problems and computing the minimum spanning tree (MST) of weighted and undirected graphs. The computing using Cellular Neural Networks (CNN) is considered for ultra-fast solutions when dealing with graphs of complex topology. As proof of concepts of the proposed method, simulations are performed on graphs of magnitude 11 and degree 8. The results obtained show the efficiency of the novel method.**

Keywords*— Cellular Neural Networks, Lagrangian Relaxation, Multivariate Optimization, NP-hard Problems, Minimum Spanning Tree

I. INTRODUCTION

The optimization issue is an essential step towards the modeling of NP-hard problems [1-3]. Example of such problems can be investigating the minimum spanning tree (MST) in a complex graph. Investigating MST is a striking problem leading to various potential applications basically in networks. Indeed, many engineering problems can be expressed by MST; examples of these problems can be: designing communication networks or electrical power systems, routing in packet networks [2-4], just to name a few. In this paper, we present a novel method for computing minimum spanning tree. The neuron-dynamics concept [5] is exploited to transform the optimization function derived with respect to a considered graph topology into sets of ordinary differential equations (ODEs). The ODEs are solved using the CNN concept. This paper is organized as follows: Section 2 deals with a brief description of the CNN model used to design the CNN simulator. Section 3 presents a general theory showing how the optimization function can be conducted by combining both objective (or cost) function with constraints and the principle of deriving sets of ODEs. Section 4 applies the general optimization theory to a specific problem which is finding the MST. This problem is formulated analytically and sets of mathematical expressions are derived which are being used to evaluate the MST. Section 5 deals with the design of the CNN simulator to solve the proposed ODEs. Section 6 shows the simulation results obtained with CNN. Various graph topologies with same magnitudes but different weights are envisaged and the MST is computed for each of them. Section 7 deals with conclusions and some open research directions.

II. CELLULAR NEURAL NETWORKS

The concept of CNN was introduced by Leon O. Chua and Lin Yang in 1988. The original idea was to use an array of simple, non-linearly coupled dynamic circuits to process in parallel large amounts of data in real time [6]. The CNN processor is built using a large array of interconnected nonlinear dynamic systems called cells. The state control CNN (SC-CNN) with a self-feedback template is modeled by (1) [7]. x_i, y_i and u_i are the state, output and input variables respectively.

$$\dot{x}_i = -x_i + \sum_{j=1}^{M} \left[\hat{\mathrm{A}}_{ij} x_j + \mathrm{A}_{ij} y_j + \mathrm{B}_{ij} u_j \right] + I_i \qquad (1)$$

The coefficients $\hat{\mathrm{A}}_{ij}$, A_{ij}, B_{ij} are the self-feedback template, feedback template and control template, respectively. The schematic model of a SC-CNN cell coupled to (M-1) neighbouring cells is depicted in Fig. 1.

The Cellular Neural Networks are particularly interesting because of the programmable nature and capabilities of implementation by VLSI on-chip technologies [7]. Major applications of CNN are found in image processing and nonlinear dynamics, just to name a few. These applications are generally modeled by ODEs and PDEs. The next development is concerned with the design of a CNN simulator to solve the NP-hard problem related to finding the minimum spanning tree in many graph topologies of magnitude 11 and degree 8.

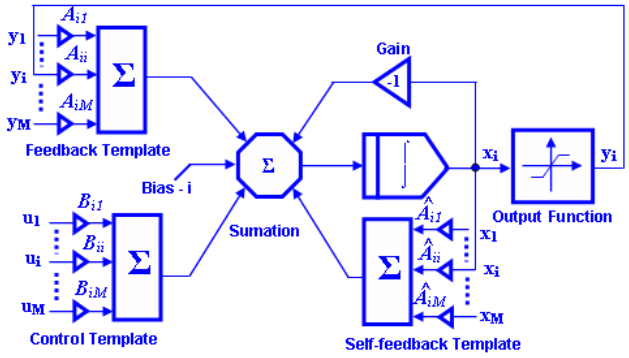

Figure 1. Structure of a SC-CNN cell coupled to (M-1) neighbouring cells

III. MODELING NP-HARD PROBLEMS BY ODES

The multivariate optimization problem is represented by an objective function subjected to constraints. These constraints make the problem NP-hard basically when their number becomes significantly important. In order to solve NP-hard problems the constraint optimization issues can be changed into unconstraint optimization issues using relaxation methods. Several methods are available to transform hard problems into simple ones. Some of these methods are based on: direct substitution, constrained variation, and Lagrangian multipliers [8], just to name a few.

The minimum spanning tree can be considered as a multivariate constrained optimization problem. This problem can be modeled by (2a), (2b) and (2c).

$$\text{Mininize} \quad f = f(X) \tag{2a}$$

$$\text{subject to}: \quad g_k(X) \le 0 \qquad k = 1,2,...,m \tag{2b}$$

$$h_j(X) = 0 \qquad j = 1,2,...,p \tag{2c}$$

Where $X = [x_1, x_2, ..., x_n]$ is the state vector of all the edges in a given graph and $f(X)$ is a scalar value materializing or representing the distance of spanning trees. During the optimization process, the value of the state variables should be attracted to the constraints defined by (2b) and (2c). This reveals the sensitive effects of constraints on the optimal state variables.

The Lagrangian multiplier is a widely used relaxation technique for integer programming problems. Some of the characteristics of the Lagrangian relaxation method are described in [8-10]. In the general case of multivariate optimization problems, the Lagrangian multiplier method converts the objective function together with all constraints into Lagrangian function by introducing a set of Lagrangian multipliers λ_k, where each λ_k corresponds to one constraint $g_k(X)$. This can be modeled by (3).

$$\tilde{L}(X,\lambda) = f(X) + \sum_{k=1}^{m} \lambda_k \times g_k(X) + \sum_{j=1}^{p} \lambda_j \times h_j(X) \tag{3}$$

where $\lambda_k \ge 0$

In order to minimize the objective function f, suitable values for the Lagrange multipliers must be calculated. In our approach, we consider these Lagrangian multipliers as Lagrangian variables. These variables are control parameters which can be monitored to minimize the objective function. When this condition is fullfilled the optimal value of the state variables are obtained showing the minimum spanning tree of a specific graph topology.

The set of ODEs is derived by applying the neuron-dynamics theorem [5] to all the partial derivatives in term of x_i and λ_k. This leads to (4a) and (4b).

$$\frac{dx_i}{dt} = -\frac{\partial \tilde{L}(x_i, \lambda_k)}{\partial x_i} \tag{4a}$$

$$\frac{d\lambda_k}{dt} = +\frac{\partial \tilde{L}(x_i, \lambda_k)}{\partial \lambda_k} \tag{4b}$$

In summary, three key steps are considered for modeling NP-hard problem by ODEs. These steps are shown in Fig. 2.

The first step is concerned with the derivation of the objective function and the modeling of constraints. The corresponding equations are proposed.

The second step deals with the establishment of the Lagrangian optimization function. This is archived by combining the objective function with all the constraints.

The third step is concerned with the derivation of the corresponding partial differential equations (PDEs). This is achieved by evaluating the partial derivatives of the Lagrangian optimization function in terms of all the independent variables. The concept of neuron-dynamics is exploited to transform the PDEs into ODEs.

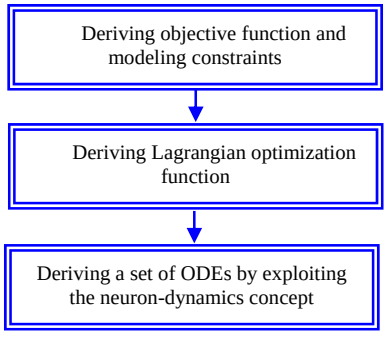

Figure 2. Key steps of modeling NP-hard problems by ODEs

IV. MST PROBLEM FORMULATION

Consider a weighted and undirected graph G = (V, E) consisting of N nodes (vertex set V) connected by e(i,j) ∈ E (edge set E), where i and j are integers. Each connectivity between node i and node j is assigned a specific real number (c_{ij}) which is a factor/coefficient proportional to the distance between two nodes.

Our focus is finding the minimum spanning tree of a given graph. To achieve this, the distance between nodes is the main factor which is being considered for findings. This distance is obtained as the sum of distances between nodes (i.e. nodes belonging to the MST). The connectivity between nodes is materialized by the state x_{ij} of each edge ($x_{ij}=1$ for edges belonging to the MST and $x_{ij}=0$ otherwise).

The distance in the graph G (V, E) can be expressed or formulated by the objective function $f(x)$ described in (5).

$$f(X)=\sum_{i=1}^{N-1}\sum_{j=i+1}^{N} c_{ij}x_{ij} \qquad \begin{cases} x_{ij}\in F=\{0,1\} \\ \forall i,j \,|\, c_{ij}\neq 0 \end{cases} \tag{5}$$

The optimization function to compute the MST is obtained from (6).

$$Q(X)=\underset{X}{\text{Minimize}}\ [\,f(X)] \tag{6}$$

Some key conditions are considered to fulfill the requirements of the problem formulation. The first condition imposes every node in the graph to belong to the spanning tree. This can be formulated by (7)

$$1-\sum_{\substack{j=1 \\ j\neq k}}^{N} x_{kj}\leq 0 \qquad \begin{cases} k=1,2,\ldots N \\ \forall j \,|\, c_{kj}\neq 0 \end{cases} \tag{7}$$

Where index k stands for every nodes in the graph. The j index stands for all the nodes connected to node k .

The second condition is avoiding cycles in the sub-graphs. This can be formulated by (8).

$$\sum_{i=1}^{N-1}\sum_{j=i+1}^{N} x_{ij}-N+1=0 \qquad \forall i,j \,|\, c_{ij}\neq 0 \tag{8}$$

The Lagrangian multiplier method is applied to combine the objective function together with constraints into the Lagrangian optimization function as described in (9)

$$\tilde{L}(x_{ij},\lambda_k)=\sum_{i=1}^{N-1}\sum_{j=i+1}^{N} c_{ij}x_{ij}+\lambda_k\left(1-\sum_{\substack{j=1 \\ j\neq k}}^{N} x_{kj}\right) + \lambda_{N+1}\left(\sum_{i=1}^{N-1}\sum_{j=i+1}^{N} x_{ij}-N+1\right) \tag{9}$$

where $\lambda_k \geq 0$

Equation (9) can be transformed into (10a), (10b) and (10c) by applying the neuron-dynamics concept.

$$\frac{dx_{ij}}{dt}=\lambda_i+\lambda_j-\lambda_{N+1}-c_{ij} \qquad \forall i,j \,|\, c_{ij}\neq 0 \tag{10a}$$

$$\frac{d\lambda_k}{dt}=1-\sum_{j=1}^{N} x_{kj} \qquad \begin{cases} k=1,2,..,N \\ \forall j \,|\, c_{kj}\neq 0 \end{cases} \tag{10b}$$

$$\frac{d\lambda_{N+1}}{dt}=\sum_{i=1}^{N-1}\sum_{j=i+1}^{N} x_{ij}-N+1 \qquad \forall i,j \,|\, c_{ij}\neq 0 \tag{10c}$$

V. SOLUTION BASED ON THE CNN CONCEPT

The CNN simulator is designed on top of SIMULINK to solve the set of ODEs in (10a), (10b) and (10c). These equations are used to evaluate the MST in the graph topology in Fig. 3a [3]. Indeed, the solutions x_{ij} are used to decide the states of connectivity of edges belonging to the MST. Three main steps are considered for the implementation of the CNN simulator:

a. The ODEs are arranged in the similar form as the state-control CNN equations (1). A comparison or identification of these equations leads to the determination of the appropriate (or corresponding) CNN templates to solve (10a), (10b) and (10c).

b. We design a CNN simulator to solve 34 ODEs with 34 variables corresponding to 22 state variables and 12 Lagrangian variables. This is achieved by using 34 CNN cells.

c. Each of the 22 state variables can take the values 1 or 0. This condition is fulfilled if we are performing in the linear portion of the CNN sigmoid function. Therefore one can perform some mathematical development to show that the state variables are accurately obtained at the output of the sigmoid function.

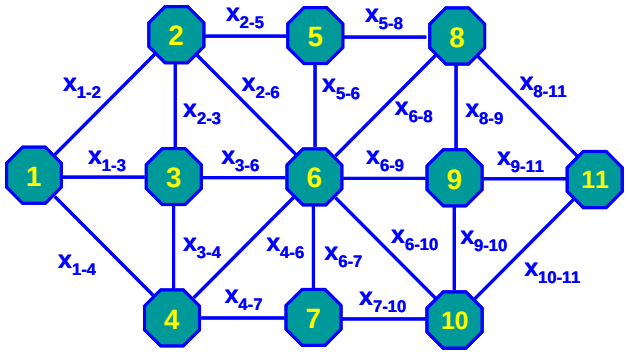

Figure 3a. A graph topology with 11 nodes

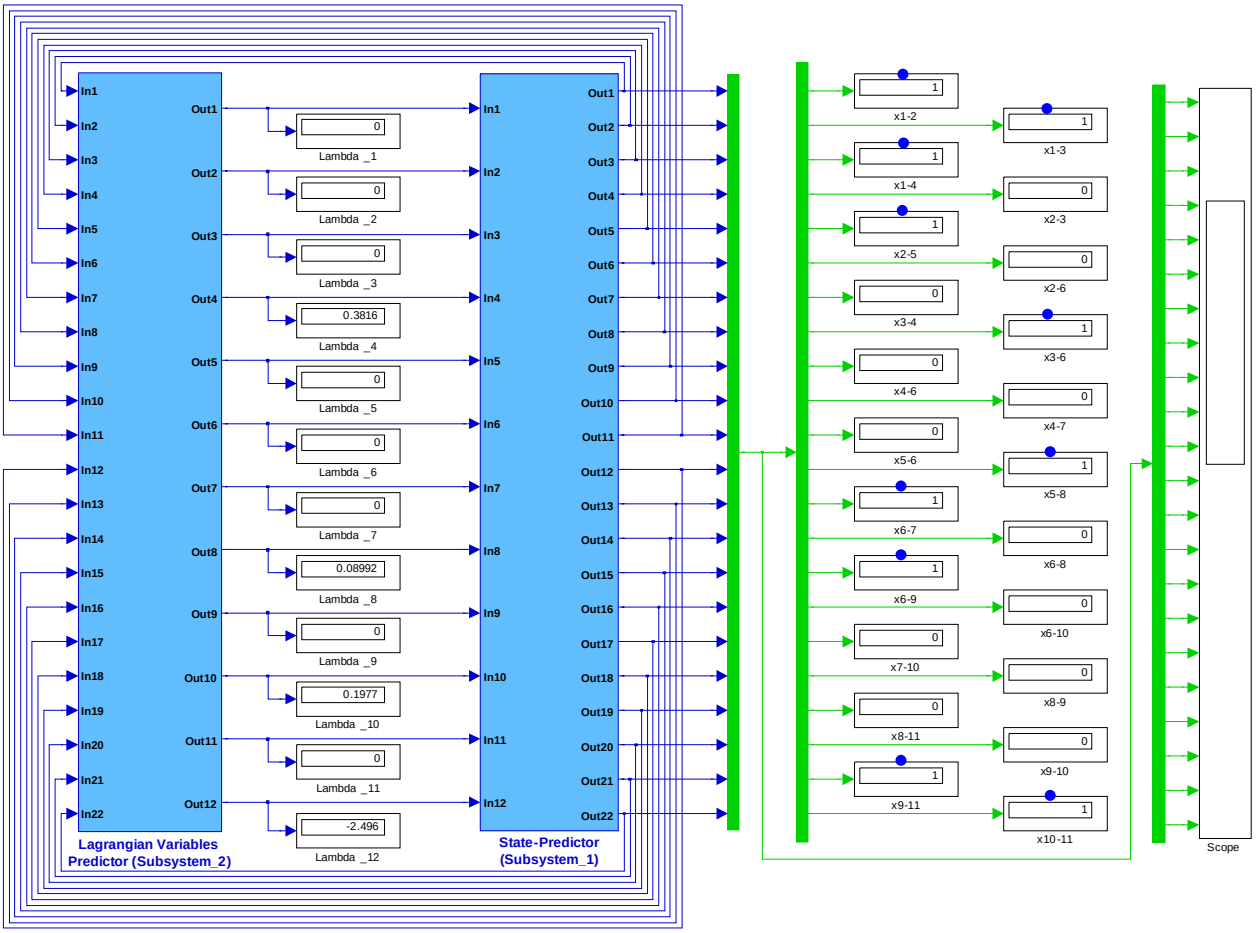

Figure 3b. SIMULINK diagram of the CNN simulator

Output result of scenario-1

* State variables' values:

$$x_{1\text{-}2} = 1,\ x_{1\text{-}3} = 1,\ x_{1\text{-}4} = 1,\ x_{2\text{-}5} = 1,\ x_{3\text{-}6} = 1,\ x_{5\text{-}8} = 1,$$

$$x_{6\text{-}7} = 1,\ x_{6\text{-}9} = 1,\ x_{9\text{-}11} = 1,\ x_{10\text{-}11} = 1$$

* Minimum Spanning Tree:

{(1, 2), (1, 3), (1, 4), (2, 5), (3, 6), (5, 8), (6, 7), (6, 9), (9, 11), (10, 11)}

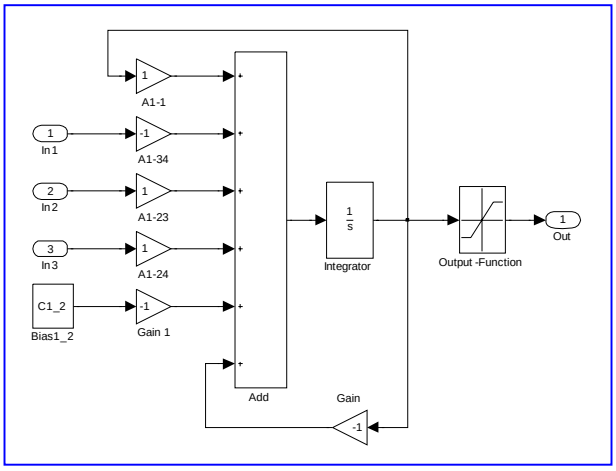

Figure 3c: Structural sub-block of the elementary CNN cell for a specific state variable

It should be worthnoticing that the condition (c) shows the advantage of using the CNN concept to determine the state variables. In fact, the sigmoid function is exploited to impose the state variables between 0 and 1. The SIMULINK diagram of the CNN simulator shown in Fig. 3b consists of two subsystems. The first subsystem is the state predictor. The second is the Lagrangian variable predictor. Each subsystem includes number of sub-blocks each of which implements the structure of the elementary cell (Fig. 3c). Each elementary cell for a specific state variable has three connections with other neighbouring cells. The weight value of each edge is the bias value of a cell.

VI. SIMULATION PRINCIPLE AND RESULTS

A. General principle

Our simulation principle is summarized in Fig. 4. Four main/key steps are considered.

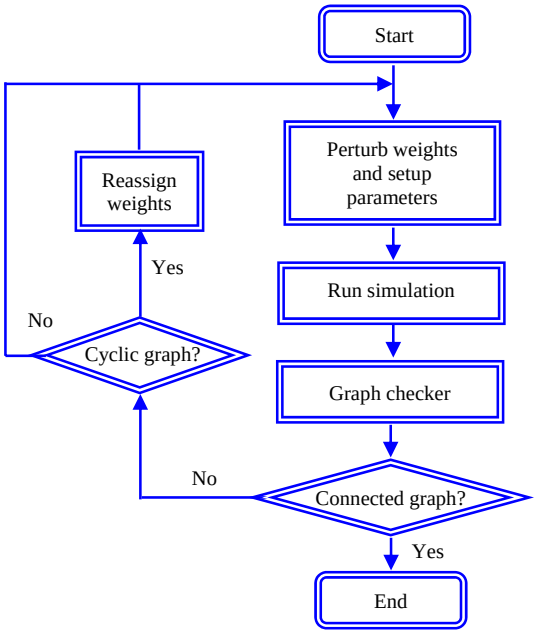

Figure 4. General principle of simulations

The first step deals with the perturbation of weights and setup parameters. This step is very important for obtaining converging simulations which manifest themselves by the iniquity of solutions (i.e. the existence of only one or a unique MST for a given graph topology). In fact, a situation can occur (depending upon the weight values assigned to a graph topology) where we have more than one MST. Another situation can occur where looking for MST leads to cyclic sub-graphs. These are two main problems we are currently facing when evaluating MST in various graph topologies. To overcome the problem related to the coexistence of many MST for the same simulation, we perform a dynamic variation of weights based on the perturbation analysis of all weight values of the graph under investigation. This perturbation analysis was shown to be efficient to decide about the final MST corresponding to a specific graph topology. Referring to the existence of cyclic sub-graphs when evaluating MST, this issue was solved by choosing one weight randomly amongst those belong to the cycle and assigning it a big weight value (or a heavy weight). This is the technique which consists to exclude a specific weight of the cycle.

The second step is concerned with the computing process of the CNN simulator. The output of CNN simulator corresponding to state variables are fed into the graph checker.

The third step deals with the determination of the valid sub-graph as solution of MST. This sub-graph is obtained from all edges corresponding to state variables which take values 1.

The fourth step deals with the decision of completing the simulation or continuing to run more times. This is based on checking information provided by the graph checker. In case of unconnected sub-graphs with or without cycles, the simulation will be re-run after perturb weights and/or reassign weight values to one edge belonging to the cycle. This scenario is materialized in Fig. 4 by a feedback loop. The simulation finished when the minimum spanning tree found or after a predefined number of iterations.

B. Simulation results

We perform simulations for finding MST trajectories on a particular weighted graph topology with magnitude 11 and degree 8. For our various simulations, three scenarios are considered. The first scenario deals with cases of weights leading to connected sub-graphs without cycles (or spanning trees) with minimum distance (see Fig. 5a). The output results give the minimum spanning tree which includes all the edges represented in bold-lines in Fig. 5a based on the values of state variables obtained from the CNN simulator. The two other scenarios are concerned with the results of unconnected sub-graphs without cycles (Fig. 5b) or with cycles (Fig. 5c).

In case of scenarios 2 and 3, the simulator need to re-run one or several times (following the process shown in Fig. 4) to obtain the final result of a minimum spanning tree. An interesting question when dealing with cyclic sub-graphs is how to break the cycle. Our approach to break the cycle is to choose one of the edges belonging to the cycle and assign it a big value of weight in order to avoid the selection of that edge during the next MST findings process. After breaking the cycle, the simulation process is the same as in scenario-1.

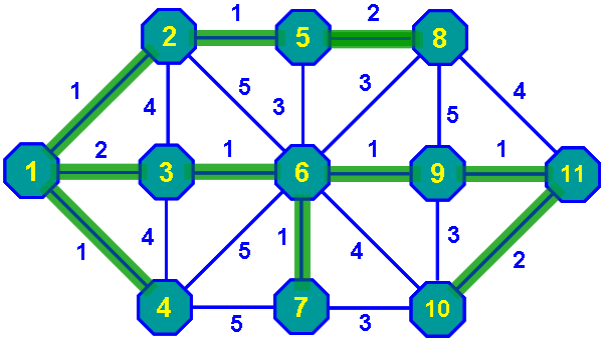

Figure 5a. Scenario-1: connected subraphs without cycles

In the Table I, it is clearly shown that the computing process starts by perturbing all weights together. This can be derived by a small value of the order of perturbation added to every weight in the Table I, each of which can be localized in Fig. 5b.

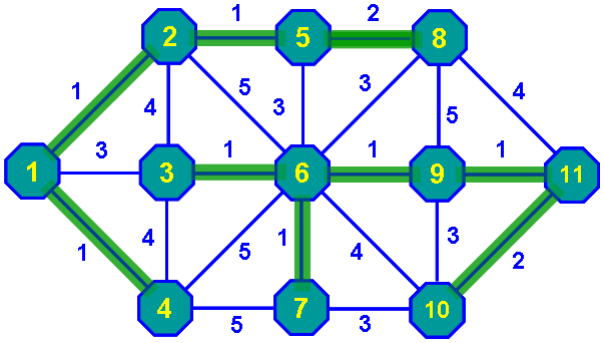

Figure 5b. Scenario-2: unconnected subgraphs without cycles

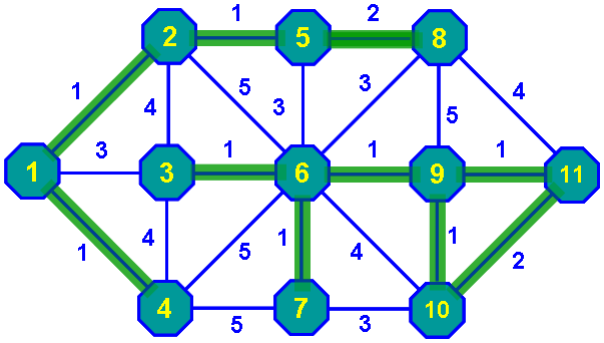

Figure 5c. Scenario-3: unconnected subgraphs with cycles

TABLE I. PERTURBED WEIGHTS OF SCENARIO-2

Node	1	2	3	4	5	6	7	8	9	10	11
1	0.0000	1.0901	3.0168	1.1827	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
2	1.0901	0.0000	4.0438	0.0000	1.1652	5.1077	0.0000	0.0000	0.0000	0.0000	0.0000
3	3.0168	4.0438	0.0000	4.0305	0.0000	1.1992	0.0000	0.0000	0.0000	0.0000	0.0000
4	1.1827	0.0000	4.0305	0.0000	0.0000	5.0156	5.0213	0.0000	0.0000	0.0000	0.0000
5	0.0000	1.1652	0.0000	0.0000	0.0000	3.0885	0.0000	2.0009	0.0000	0.0000	0.0000
6	0.0000	5.1077	1.1992	5.0156	3.0885	0.0000	1.1924	3.1550	1.1635	4.0169	0.0000
7	0.0000	0.0000	0.0000	5.0213	0.0000	1.1924	0.0000	0.0000	0.0000	3.0800	0.0000
8	0.0000	0.0000	0.0000	0.0000	2.0009	3.1550	0.0000	0.0000	5.1737	0.0000	4.1600
9	0.0000	0.0000	0.0000	0.0000	0.0000	1.1635	0.0000	5.1737	0.0000	3.0520	1.0863
10	0.0000	0.0000	0.0000	0.0000	0.0000	4.0169	3.0800	0.0000	3.0520	0.0000	2.1821
11	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	4.1600	1.0863	2.1821	0.0000

Results of CNN simulator using perturbed weights in Table I at run#2 corresponding to the trajectory of the MST

Edge(i,j): {(1,2), (1,3), (1,4), (2,5), (3,6), (5,8), (6,7), (6,9), (9,11), (10,11)}

TABLE II. SCENARIO-3.1: PERTURBED WEIGHTS, W(10,11) = 100

Node	1	2	3	4	5	6	7	8	9	10	11
1	0.0000	1.4505	3.0838	1.9133	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
2	1.4505	0.0000	4.2290	0.0000	1.8258	5.5383	0.0000	0.0000	0.0000	0.0000	0.0000
3	3.0838	4.2290	0.0000	4.1524	0.0000	1.9961	0.0000	0.0000	0.0000	0.0000	0.0000
4	1.9133	0.0000	4.1524	0.0000	0.0000	5.0782	5.1067	0.0000	0.0000	0.0000	0.0000
5	0.0000	1.8258	0.0000	0.0000	0.0000	3.4427	0.0000	2.0046	0.0000	0.0000	0.0000
6	0.0000	5.5383	1.9961	5.0782	3.4427	0.0000	1.9619	3.7749	1.8173	4.0844	0.0000
7	0.0000	0.0000	0.0000	5.1067	0.0000	1.9619	0.0000	0.0000	0.0000	3.3998	0.0000
8	0.0000	0.0000	0.0000	0.0000	2.0046	3.7749	0.0000	0.0000	5.8687	0.0000	4.8001
9	0.0000	0.0000	0.0000	0.0000	0.0000	1.8173	0.0000	5.8687	0.0000	1.2599	2.4314
10	0.0000	0.0000	0.0000	0.0000	0.0000	4.0844	3.3998	0.0000	1.2599	0.0000	100.91
11	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	4.8001	2.4314	100.91	0.0000

Results of CNN simulator using perturbed weights in Table II at run#1 corresponding to the trajectory of the MST

Edge(i,j): {(1,2), (1,3), (1,4), (2,5), (3,6), (5,8), (6,7), (6,9), (9,10), (9,11)}

TABLE III. SCENARIO-3.2: PERTURBED WEIGHTS, W(9,11) = 100

Node	1	2	3	4	5	6	7	8	9	10	11
1	0.0000	1.1818	3.2638	1.1361	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
2	1.1818	0.0000	4.1455	0.0000	1.5797	5.5499	0.0000	0.0000	0.0000	0.0000	0.0000
3	3.2638	4.1455	0.0000	4.8693	0.0000	1.1450	0.0000	0.0000	0.0000	0.0000	0.0000
4	1.1361	0.0000	4.8693	0.0000	0.0000	5.8530	5.3510	0.0000	0.0000	0.0000	0.0000
5	0.0000	1.5797	0.0000	0.0000	0.0000	3.6221	0.0000	2.4018	0.0000	0.0000	0.0000
6	0.0000	5.5499	1.1450	5.8530	3.6221	0.0000	1.5132	3.0760	1.2399	4.1839	0.0000
7	0.0000	0.0000	0.0000	5.3510	0.0000	1.5132	0.0000	0.0000	0.0000	3.2400	0.0000
8	0.0000	0.0000	0.0000	0.0000	2.4018	3.0760	0.0000	0.0000	5.1233	0.0000	4.0497
9	0.0000	0.0000	0.0000	0.0000	0.0000	1.2399	0.0000	5.1233	0.0000	1.4173	100.90
10	0.0000	0.0000	0.0000	0.0000	0.0000	4.1839	3.2400	0.0000	1.4173	0.0000	2.9448
11	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	4.0497	100.90	2.9448	0.0000

Results of CNN simulator using perturbed weights in Table III at run#1 corresponding to the trajectory of the MST

Edge(i,j): {(1,2), (1,4), (2,5), (3,6), (5,8), (6,7), (6,8), (6,9), (9,10), (10,11)}

TABLE IV. SCENARIO-3.3: PERTURBED WEIGHTS, W(9,10) = 100

Node	1	2	3	4	5	6	7	8	9	10	11
1	0.0000	1.6491	3.7317	1.4509	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
2	1.6491	0.0000	4.6477	0.0000	1.2963	5.7447	0.0000	0.0000	0.0000	0.0000	0.0000
3	3.7317	4.6477	0.0000	4.5470	0.0000	1.1890	0.0000	0.0000	0.0000	0.0000	0.0000
4	1.4509	0.0000	4.5470	0.0000	0.0000	5.6888	5.3685	0.0000	0.0000	0.0000	0.0000
5	0.0000	1.2963	0.0000	0.0000	0.0000	3.1835	0.0000	2.7802	0.0000	0.0000	0.0000
6	0.0000	5.7447	1.1890	5.6888	3.1835	0.0000	1.6256	3.0811	1.9294	4.4868	0.0000
7	0.0000	0.0000	0.0000	5.3685	0.0000	1.6256	0.0000	0.0000	0.0000	3.4359	0.0000
8	0.0000	0.0000	0.0000	0.0000	2.7802	3.0811	0.0000	0.0000	5.7757	0.0000	4.3063
9	0.0000	0.0000	0.0000	0.0000	0.0000	1.9294	0.0000	5.7757	0.0000	100.44	1.5085
10	0.0000	0.0000	0.0000	0.0000	0.0000	4.4868	3.4359	0.0000	100.44	0.0000	2.5108
11	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	4.3063	1.5085	2.5108	0.0000

Results of CNN simulator using perturbed weights in Table IV at run#2 corresponding to the trajectory of the MST

Edge(i,j): {(1,2), (1,4), (2,5), (3,5), (5,8), (6,7), (6,8), (6,9), (9,11), (10,11)}

During our various computations, we obtain a cyclic sub-graph when finding the MST, especially when the graphs have many equal or close values of weights. This cyclic sub-graph is materialized in Fig. 5c where it is clearly shown that the cycle in the sub-graph is composed by the following sequence of edges {(9,10), (10,11), and (10,11)}. In this case, one of these edges {W (9,10), W (10,11), or W (10,11)} is assigned a heavy weight (value = 100). This leads to the trajectory of MST after one or two runs as shown together with perturbed weights in Tables II, III and IV. In these scenarios, the summation of weights of sub-graphs specified by state variables solved by CNN simulator can change randomly with small variation around the distance of the real MST.

During our various numerical simulations, it has been found that the convergence time of state variables depend significantly on the values of weights. This convergence time corresponds to the duration of the transient phase. In fact, each of the 34 ODEs can show different convergence times. Therefore, the convergence time of the global system (set of 34 ODEs) is chosen to be equal to the maximal convergence time of state variables. The convergence time of full system is corresponding to the convergence time of the state variable x_{13} as drawn in Fig. 6.

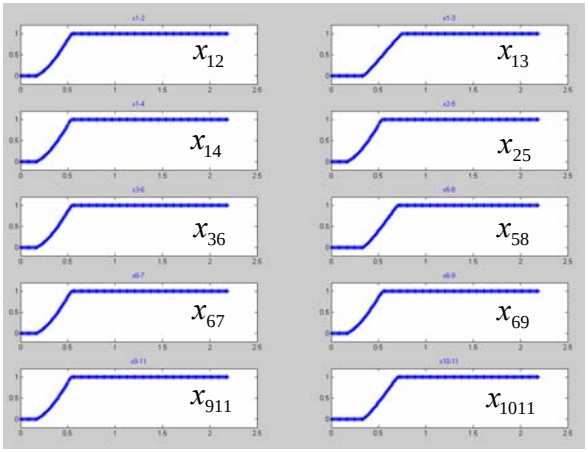

Figure 6. Convergence times of state variables (scenario-1)

During our various simulations, it has been found that all the state variables corresponding to the edges belonging to the MST which take the same value of weight will have the same convergence time. For scenario-1 (Fig. 5a), six state variables corresponding to the edges {(1,2), (1,4) , (2,5) , (3,6) , (6,7) , (6,9), and (9,11)} which have weight values 1, as well as three state variables corresponding to the edges {(1,3), (5,8), and (10,11)} which have weight values 2 show the same convergence times (see Fig. 6). Among all the state variables belonging to the MST, the state variables of the edges with greater value of weight will take more time to converge to value 1 (i.e. the permanent value of state variables belonging to the MST).

VII. CONCLUSIONS

This paper has shown the good potentiality of the computing based on the CNN paradigm/concept to solve NP-hard problems. This paradigm has been applied on a graph topology of magnitude 11 and degree 8 to find the minimum spanning tree. The proposal method and simulation principle can be very flexible to real issues where random weighted graphs are involved. One issue need to be further considered is how to model the general constraints for removing completely cycles and/or impose the findings of connected sub-graphs. The implementation of the CNN simulator to compute stochastic MST on real hardware platforms by combining powerful features of DSP and FPGA can also be the next developments to explore the advantages of the proposal methods. This can increase the stability of the simulator and speed-up the convergence times.

ACKNOWLEDGMENT

Dr. Do Trong Tuan would like to express the sincere thanks to the Austrian Federal Ministry for Science and Research (BMWF) for the OEAD scholarship of post-doctoral research program (from 15.08.2008 to 14.08.2009) at Institute for Smart-Systems Technologies, University of Klagenfurt, AUSTRIA.

REFERENCES

[1] D. T. Tuan, J. C. Chedjou, K. Kyamakya, On Transforming Graph Theoretical Problems into Optimization Problems and Solution using CNN-based Analog Computing, *ISTET Conference, Germany,* 2009.

[2] Bang Yewu and Kun Mao Chao, Spanning Trees and Optimization Problems, *Chapman and Hall/CRC Press*, 2004.

[3] V. K: Balakrishnan, Theoy and Problems of Graph Theory, *Mcgraw-Hill*, 1997

[4] J. A. Bondy and U. S. R Murty, Graph Theory with Applications, *North-Holland Press*, 1976

[5] Peter B. Luh et al, Lagrangian Neural Networks for Job Shop Scheduling, *IEEE Trans. On Robotics-Automation*, Vol.16, 78-88.

[6] L.O Chua and L. Yang, Cellular Neural Networks: Theory, *IEEE Trans. On Circuits and Systems*, Vol.35, 1257-1272.

[7] G. Manganaro, P. Arena, L. Fortuna, Cellular Neural Networks: Chaos, Complexity & VLSI Processing, *Springer,* 1999, 44-45.

[8] G. Singiresu Rao, Engineering Optimization: Theory and Practice, *Wiley-Interscience Pulication*, 1996, 80-112.

[9] M. Guignard. Lagrangian relaxation, *TOP*, Vol11, No2, 151-228.

[10] M.L. Fisher, The lagrangian relaxation method for solving integer programming problems, *Mag. Science*, Vol27, 1-18