Do Trong Tuan[1], Jean Chamberlain Chedjou[2], Kyandoghere Kyamakya[2]

Hanoi University of Technology (1), University of Klagenfurt (2)

On Transforming Graph Theoretical Problems into Optimization Problems and Solution using CNN-based Analog Computing

Abstract. *We present a novel method for transforming graph theoretical problems into optimization problems and calculating shortest path/walk in both directed and undirected graphs. Analog Computing based on Cellular Neural Networks (CNN) paradigm is carried out to derive ultra-fast solutions when dealing with graphs of complex topologies. As proof of concepts of the proposed method, simulations are performed on graphs of magnitude 11 and the results obtained show the efficiency of the novel method. The proposed method is challenging as it can be extended to solving scheduling issues in real-world scenarios which are NP-hard problems.*

Keywords. *Cellular Neural Network, Lagrangian Relaxation, Multivariable Optimization, NP-hard Problem, Shortest Walk/Path Problem*

1. Introduction

The optimization issue is an essential step towards the modeling of NP-hard problems. Graphs with complex topologies are good prototypes of such problems. In real-life, shortest path problems can be exploited in many applications such as: message routing in communication networks, directions finding between two physical locations and controlling vehicular traffic flows through congested cities [1,2] just to name a few. In this paper, we present a novel method for transforming graph theoretical problem into optimization problems . The neuron-dynamics concepts [3] is exploited to transform the optimization function derived with respect to the considered graph topology into sets of ordinary differential equations (ODEs). The ODEs are solved using the CNN paradigm. This paper is organized as follows: Section 2 deals with a brief description of the CNN model used to design the complete CNN simulator. Section 3 presents a general theory showing how the optimization function can be conducted by combining both objective (or cost) function and constraints and the way a set of ODEs is derived. Section 4 applies the general optimization theory to a specific problem which is finding shortest path/walk. This problem is formulated analytically and sets of mathematical expressions are derived as models. Section 5 deals with the design of the complete CNN simulator to solve the proposed ODEs. Section 6 shows the simulation results obtained with the CNN. Various graph topologies with same magnitudes but different weights are envisaged and the shortest walk calculation is performed for each graph topology. The results obtained using our approach are compared with the results obtained by the linear programming method using the built-in function (linprog) in matlab. Section 7 deals with conclusions and some open research directions.

2. 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 [4]. The CNN processor is built using a large array of interconnected nonlinear dynamic systems called cells. The state control CNNs with a self-feedback template is modeled by Eq. (1) [5]. x_i, y_i and u_i are the state, output and input variables respectively.

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

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 state-control CNN (SC-CNN) cell coupled to (M-1) neighbour cells is depicted in Fig. 1.

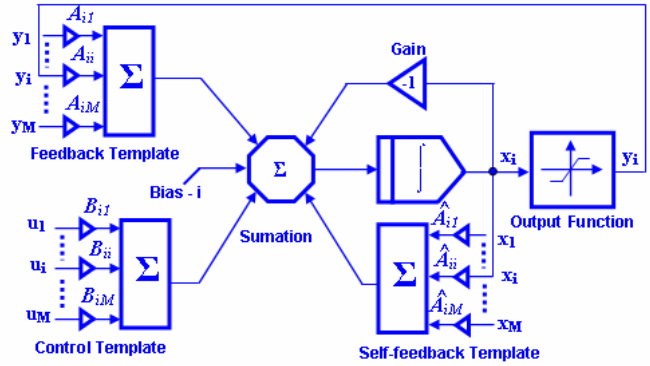


Fig. 1. Structure of a SC-CNN Cell coupled to (M-1) Neighbours

The Cellular Neural Networks are particularly interesting because of the programmable nature and capabilities of implementation by VLSI on-chip technologies [5]. Major applications of CNN are found in performing image processing tasks and solving non-linear dynamic systems, just to name a few. These applications are generally modeled by ODEs and PDEs. The next development is concerned with the design of a complete CNN simulator/processor (which elementary cells are modeled by Eq. (1)) to solve the optimization problem related to finding a shortest walk in many graph topologies of magnitude 11.

3. Transformation method

The multivariable optimization problem is represented by an objective function subjected to constraints. These constraints make the problem NP-hard basically when their number became 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: direct substitution, constrained variation, and Lagrange multipliers [6], just to name a few.

The shortest walk problem can be considered as a multivariable constrained optimization problem. This type of problems can be modeled by Eqs. (2).

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

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

Where $X=[x_1,x_2,...,x_n]$ is the state vector of all the edges in the given graph and $f(X)$ is a scalar value which is the distance of the walk from a source node to a destination node. During the optimization process, the value of the state variables should be attracted to the constraints defined by Eq. (2b).

The Lagrange multiplier is a widely used relaxation technique for integer programming problems. Some of the characteristics of the Lagrangian relaxation method are described in [6-8]. In the general case of multivariable optimization problems with n variables and m constraints, the Lagrange multiplier method converts the objective function together with all constraints into Lagrange function by introducing a set of Lagrange multipliers λ_k, where each λ_k corresponds to one constraint $g_k(X)$. This conversion is modeled by Eq. (3).

$$\tilde{L}(x_1,x_2,..,x_i,\lambda_1,\lambda_2,...,\lambda_k)= f(X)+\lambda_1\times g_1(X)+\lambda_2\times g_2(X)+...+\lambda_k\times g_k(X) \quad (3)$$

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

The set of ODEs is derived by exploiting the neuron-dynamics concept [3] to all the partial derivatives of $\tilde{L}(x_i,\lambda_k)$ over the state variables x_i and the Lagrangian variables λ_k as stated in Eqs. (4-5).

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

$$\frac{d\lambda_k}{dt}=+\frac{\partial\tilde{L}(x_i,\lambda_k)}{\partial\lambda_k} \quad (5)$$

Eqs.(4-5) are ODEs which can be identified as the state equations of an autonomous CNN (with zero-input) made-up of M cells in order to make them solvable by the CNN paradigm.

In summary, the method to transform optimization problem into a set of first order differential equations is expressed in Fig. 2.

The first step deals with data initialization and preprocessing of weights. We specify the source node, the destination node, intermediate nodes and determine a valid set of edges. We classify the input set of edges (set E) into two subsets. One is the valid weight subset E^v and the other E^i is the violating subset of weights. All the directed edges entering the source node or leaving the destination node are violating edges and therefore belong to set E^i. Thus, $E^v = E\backslash E^i$. We remove all the state variables with respect to elements of E^i by setting the values of that state variables equal to zero. By removing violating state variables from the objective function and constraint equations we decrease the number of equations and the complexity of problem also decreases consequently.

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

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

The fourth step is concerned with the derivation of the corresponding partial differential equations (PDEs). This is archived 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.

The last step is solving the ODEs by the CNN paradigm.

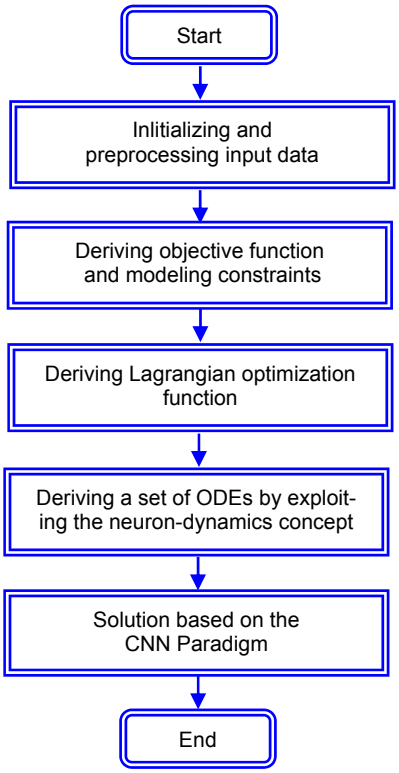


Fig. 2. Complete steps for solving NP-hard problems with Cellular Neural Network (CNN)

4. Shortest Walk Optimization Problem Formulation

Consider a weighted 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 non-negative real number c_{ij}, which is a factor/coefficient proportional to the distance between two nodes. An edge can be directed or undirected.

Our focus is finding the shortest walk from the source node (s) to the destination node (t). This walk is obtained as the sum of distances between nodes (i.e. nodes belonging to the shortest walk). The connectivity between nodes is materialized by the state x_{ij} of each edge (x_{ij} = 1 for edges belonging to the shortest walk and x_{ij} = 0 otherwise).

The walks in the graph G(V,E) can be expressed/formulated by the objective function f(X) described in Eq. (6).

$$f(X)=\sum_{i=1}^{N}\sum_{\substack{j=1\\ j\neq i}}^{N}c_{ij}x_{ij} \quad \begin{cases} x_{ij}\in F=\{0,1\} \\ \forall i,j \,|\, c_{ij}\neq 0 \end{cases} \quad (6)$$

The optimization function to calculate the shortest walk is obtained from Eq. (7).

$$Q(X)=\underset{X}{\text{Minimize}}\ [f(X)] \quad (7)$$

Some key conditions are considered to fulfill the requirements of the problem formulation. The first condition is avoiding loops and/or cycles in the walk. This can be formulated by Eq. (8)

$$(8)\quad \sum_{\substack{i=1\\ i\neq k}}^{N} x_{ik} - \sum_{\substack{j=1\\ j\neq k}}^{N} x_{kj} = 0 \qquad \begin{cases} k = 1,2,\ldots N, k \neq s,t \\ \forall i \left| c_{ik} \neq 0, \forall j \right| c_{kj} \neq 0 \end{cases}$$

Where k stands for intermediate nodes. The index i stands for nodes of the incoming edges (to node k) while j stands for nodes of the outgoing edges (from node k).

The second condition imposes nodes (s) and (t) to belong to the shortest walk. This can be formulated by Eqs.(9).

$$(9a)\quad \sum_{\substack{j=1\\ j\neq s}}^{N} x_{sj} - 1 = 0, \quad \forall j \left| c_{sj} \neq 0 \right.$$

$$(9b)\quad \sum_{\substack{i=1\\ i\neq t}}^{N} x_{it} - 1 = 0, \quad \forall i \left| c_{it} \neq 0 \right.$$

The Lagrange multiplier method is applied to combine the objective function and constraints into the Lagrangian optimization function as described in Eq. (10)

$$\widetilde{L}(x_{ij}, \lambda_k) = \sum_{i=1}^{N}\sum_{\substack{j=1\\ j\neq i}}^{N} c_{ij} x_{ij} + \lambda_s \left(\sum_{\substack{j=1\\ j\neq s}}^{N} x_{sj} - 1 \right) + \lambda_t \left(\sum_{\substack{i=1\\ i\neq t}}^{N} x_{it} - 1 \right) + \sum_{\substack{k=1,2,..,N\\ k\neq s,t}} \lambda_k \left(\sum_{\substack{i=1\\ i\neq k}}^{N} x_{ik} - \sum_{\substack{j=1\\ j\neq k}}^{N} x_{kj} \right) \quad \forall i,j \left| e(i,j) \in E^v \right. \qquad (10)$$

Eq. (10) can be transformed into Eqs. (11) by applying the neuron-dynamics concept.

$$(11a)\quad \frac{dx_{sj}}{dt} = -\left(\lambda_s + \lambda_j + c_{sj}\right) \quad \forall j \left| \; c_{sj} \neq 0 \right.$$

$$(11b)\quad \frac{dx_{ij}}{dt} = -\left(\lambda_j - \lambda_i + c_{ij}\right) \quad i \neq s, \forall i,j \left| \; c_{ij} \neq 0 \right.$$

$$(11c)\quad \frac{d\lambda_s}{dt} = \sum_{\substack{j=1\\ j\neq s}}^{N} x_{sj} - 1 \qquad \forall j \left| c_{sj} \neq 0 \right.$$

$$(11d)\quad \frac{d\lambda_t}{dt} = \sum_{\substack{i=1\\ i\neq t}}^{N} x_{it} - 1 \qquad \forall i \left| c_{it} \neq 0 \right.$$

$$(11e)\quad \frac{d\lambda_k}{dt} = \sum_{\substack{i=1\\ i\neq k}}^{N} x_{ik} - \sum_{\substack{j=1\\ j\neq k}}^{N} x_{kj} \qquad \begin{cases} k = 1,2,\ldots N, k \neq s,t \\ \forall i \left| c_{ik} \neq 0, \forall j \right| c_{kj} \neq 0 \end{cases}$$

5. Solution based on the CNN paradigm

The complete CNN simulator is designed on top of Simulink to solve the set of ODEs in Eqs. (11). These equations correspond to the modeling of the particular scenarios with respect to the topology illustrated in Fig. 3a [9]. Three main steps are considered for the design of the CNN Simulator:

a. The ODEs (Eqs. (11)) are arranged in the similar form as the state-control CNN equations (Eq. (1)). A comparison/identification of these equations leads to the determination of the appropriate (or corresponding) CNN templates to solve Eqs. (11).
b. The designed CNN simulator solves 30 ODEs with 30 variables corresponding to 19 state variables and 11 Lagrangian variables. Therefore the complete CNN simulator is made-up of 30 cells.
c. Each of the 19 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.

It should be worthnoticing that the condition (c) shows the advantage of using the CNN paradigm 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 complete CNN simulator shown in Fig. 4a consists of two subsystems: (1) the state predictor subsystem and (2) the Lagrangian subsystem. Each subsystem includes number of subblocks each of which implements the structure of the elementary cell (Fig 4.b). Each CNN cell for a specific state variable has two connections with other neighbouring cells. The weight value of each edge is the bias value of a cell.

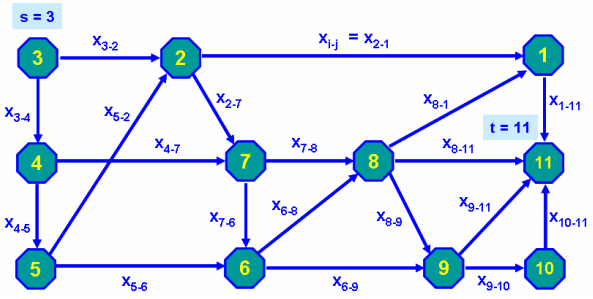


Fig. 3a. A graph topology with 11 nodes

6. Simulation results

We perform simulations for finding shortest walk trajectories on a particular weighted graph topology with magnitude of 11. During our various simulations, three scenarios are considered. The first two scenarios deal with a directed graph topology with different weight values (see Fig. 3b and Fig. 5a). The third scenario is concerned with an undirected graph topology with the same weight values as in the scenario-2 (see Fig. 5b).

The output results give the shortest walk from the source node (s=3) to the destination node (t=11). The results of the shortest walk trajectory findings are represented in bold-lines in Figs. (3a, 5a, 5b) based on the values of state variables obtained from the CNN simulator.

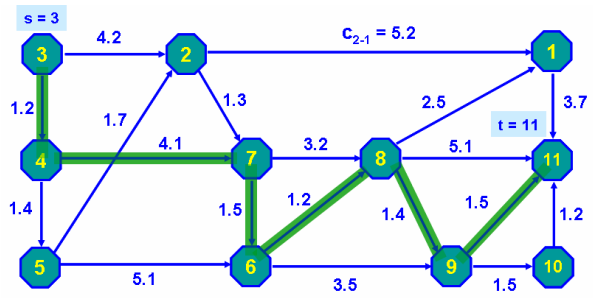
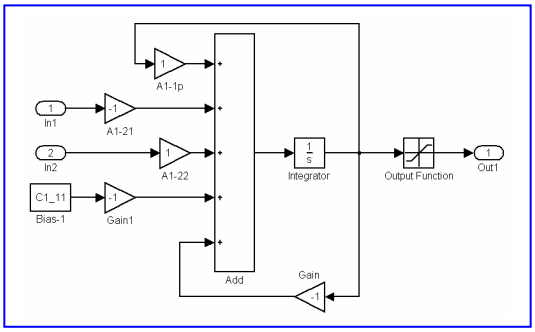


Fig. 3b. Weighted directed graph topology: scenario-1

Fig. 4a. SIMULINK diagram of the complete CNN simulator

Fig. 4b. Structural subblock of the elementary CNN cell for a specific state variable



Output result for directed graph of scenario-1

State variables' values:

$x_{3-4} = 1$, $x_{4-7} = 1$, $x_{7-6} = 1$, $x_{6-8} = 1$,

$x_{8-9} = 1$, $x_{9-11} = 1$

* Shortest walk:

3 ⇨ 4 ⇨ 7 ⇨ 6 ⇨ 8 ⇨ 9 ⇨ 11

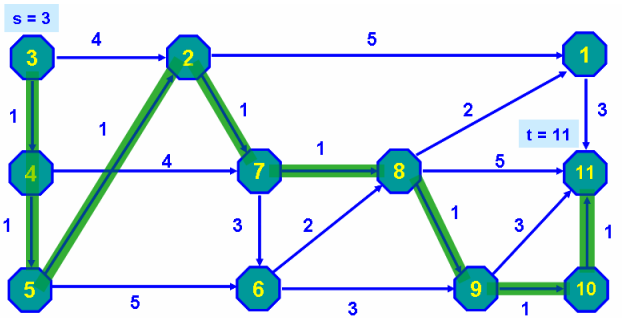


Fig. 5a. Weighted directed graph topology: scenario-2

In case of undirected graph, each undirected edge is transformed into two directed edges (incoming and outgoing edges) having the same weight as that of the undirected edge. After this transformation, the principle for finding shortest walk is the same as in the case of directed graph.

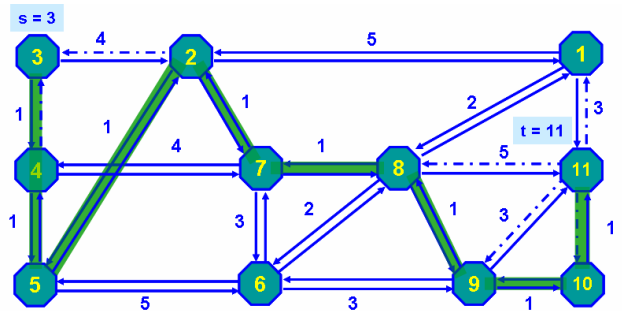


Fig. 5b. Weighted undirected graph topology: scenario-3

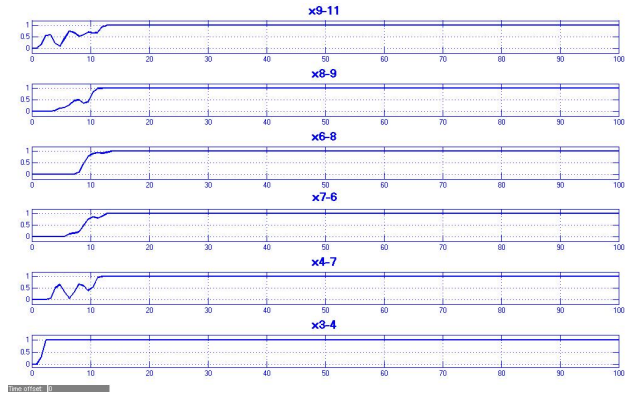


Fig. 6. Convergence time of state variables w.r.t scenario-1

During our various numerical simulations, it has been found that the convergence time of state variables depends significantly on both the number of edges included in the shortest walk and the values of weights. This convergence time corresponds to the duration of the transient phase. It has been found that each of the 30 ODEs shows different convergence times. Therefore, the convergence time of the global system (set of 30 ODEs) is chosen to be equal to the maximal convergence time of state variables. This can clearly be drawn in Fig. 6, where the convergence time of the full system is t ≈ 4.942 ms corresponding to the convergence time of state variable $x_{6\text{-}8}$. The convergence time of state variables in the proposed method changes slightly around the value 5 ms compared to 25 ms by using the linprog (built-in) function in Matlab. These results were obtained on the same computer (Pentium 4 - 2.0 GHz). The convergence time also depends on the total number of nodes of the graph as illustrated in Fig. 7. When the complexity of the graph topology increases the convergence time increases significantly when using the linear programming method. However, when the complexity of the graph topology increases the convergence time does not change significantly when simulating with the CNN paradigm. This result confirms that the good potentiality of CNN to solve NP-hard problems.

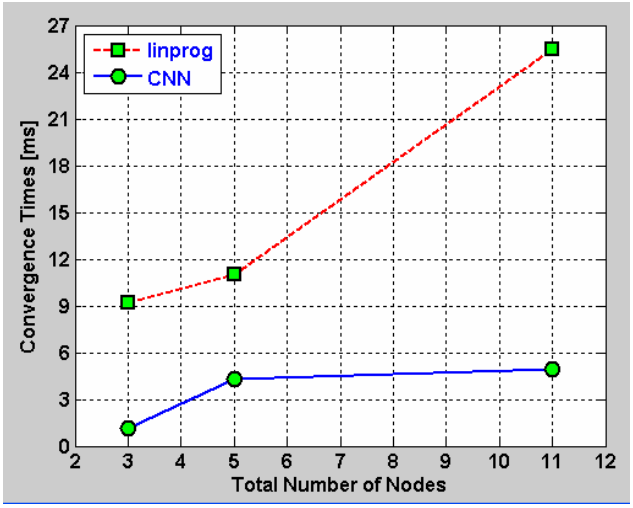


Fig. 7. Comparison of the convergence time: CNN-based method vs. Linear Programming method.

7. Conclusions

This paper has shown the good potentiality of the computing based on the CNN paradigm to solve NP-hard problems. This paradigm has been applied on a graph topology of magnitude 11; it has been shown to be faster than the linear programming method to analyze the shortest walk/path problem. This method opens promising ideas for potential applications where high speed real-time routing and scheduling issues are involved. Another interesting research question could be the design of the CNN simulator on FPGA to solve NP-hard problems. Such a concept might speed-up the convergence time, increase the accuracy of results and improve the stability of computing. This stability was shown to be very sensitive when dealing with graphs of complex topologies.

REFERENCES

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

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

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

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

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

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

[7] M Guignard. Lagrangian relaxation, *TOP*, Vol 11, No2, 151-228

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

[9] Gary Chartrand and Ping Zhang, Introduction to Graph Theory, *McGraw-Hill Companies*, 2005, 134

***Authors**:*

Dr. Do Trong Tuan, Faculty of Electronics & Telecommunications, Hanoi University of Technology, VIETNAM. He is carrying out the post-doctoral research program, which is supported by the Austrian Federal Ministry for Science and Research (BMWF), at Institute for Smart-Systems Technologies, University of Klagenfurt, AUSTRIA.
*Email: Tuan.DoTrong@uni-klu.ac.at / dotuan@*mail.hut.edu.vn
Prof. Jean Chamberlain Chedjou, Dr. Eng., Institute for Smart-Systems Technologies, University of Klagenfurt, AUSTRIA.
Email: Jean.Chedjou@uni-klu.ac.at
Prof. Kyandoghere Kyamakya, Dr. Eng., Institute for Smart-Systems Technologies, University of Klagenfurt, AUSTRIA.
Email: Kyandoghere.Kyamakya@uni-klu.ac.at