QOS-AWARE SWITCH-FABRIC MODELLING IN MATLAB ENVIRONMENT

P. Rajmic, K. Molnár

Department of Telecommunications, Faculty of Electrical Engineering and Communication,

Brno University of Technology, Czech Republic

Abstract

The paper deals with a neural network controlled switch fabric with frame prioritization support. The impact of priority levels on the functionality and efficiency of this switch fabric was simulated in MATLAB environment. Implementation details and GUI related solutions together with the simulation results are published in the paper.

1 Introduction

Our results in designing a neural network controlled switch fabric with priority support were published in [1,6]. The analysis of a switch fabric model, focused on the impact of the amount of priority levels on the system, has been evaluated and presented in them. This paper summarizes the results and in particular it describes the MATLAB environment created for the extensive analysis.

The paper is organized in the following way: First, the priority switching problem is introduced from the mathematical point of view, then the switch fabric model used during our analysis is presented. The fourth chapter describes the requirements on priority level analysis, introduces the experiment and gives main results. The fifth chapter presents the MATLAB environment used, describes its functions and possibilities.

2 Priority switching

In the case of classical data switches all the frames are of the same priority. This means that a classical switch is not able to process selected frames prior to processing others. The processing order of the frames is derived from their receiving order. This property has a very undesirable effect on real-time network applications. In the case of such an application, correct timing is essential and blocking the frames in the switch fabric can cause an incorrect function of the application.

Modern multimedia applications require the network to support Quality of Service (QoS). The physical realization of QoS support in a switch can be realized by assigning different priority levels to the frames received. The assignment procedure depends on the QoS technology implemented and is beyond the of scope of this work. Based on the assigned priority the switch can decide which frames must be processed first. Since the switch has usually several ports, the decision process is quite complex. During our work a switch with a crossbar switch fabric and input buffers was modeled.

2.1 Mathematical model of the switch

The input buffer of each port of the switch can be described by a vector. Each element of the vector corresponds to one output port in the switch. This means that the number of elements in the vector is equal to the number of ports. The first element corresponds to the first output port, the second element to the second one, etc. The value of the elements is equal to the priority of the frame that must be forwarded to this port. If there is no frame for the given output port, the element will be equal to the lowest priority level.

For example, if 0 corresponds to the highest and 255 to the lowest priority, vector (0, 5, 255, 255) means that in a four-port switch there is buffer that contains a frame with priority 0, which must be forwarded to the first output port, and another frame with priority 5, which must be forwarded to the second port. There are no frames for the third and fourth output port. In this example there are 256 priority levels. The impact of the amount of priority levels on the efficiency is the main interest of this work.

Each port of the switch can be described in this way. Collecting these vectors we get a matrix expressing the recent state of all buffers. This matrix will be used for our optimization process and it will be called *optimization matrix* and marked **C**.

2.2 The optimization process

The aim of the optimization process is to find the optimal combination of frames that can be sent out through the output ports. This means that we must derive from the optimization matrix a *configuration matrix* describing the optimal configuration (on and off states) of the switches in the switch fabric. The configuration matrix can be thought of as a filter matrix containing elements with value 1 in the places corresponding to the selected elements in the optimization matrix and value 0 for all the other elements. It is important to realize that at one time only one frame can be forwarded from one input to one output. Thus, the configuration matrix must contain just one element with value 1 (on-state of the switch) in each row and each column. These limitations express the confinement criteria of the priority switching problem.

There are several combinations fulfilling these confinement criteria. From this set of valid solutions we must select the best one. If 0 is assigned to the highest priority and larger values correspond to lower priorities, it is considered to be the best solution when the sum of the selected priority values is minimal.

Works [2] and [3] contain useful information on how to express confinement criteria so that they may be suitable for processing by the Hopfield neural network.

The Hopfield neural network is based on solving an iteration process. In our case the result of this iteration process is a configuration matrix, containing only values 0 and 1 and fulfilling the previous confinement criteria. This configuration matrix is the final state of the neural network *state matrix* **V** updated in each iteration cycle. Before the first iteration step this state matrix is generated randomly.

The state matrix **V** can be transformed into a *state vector* **v** using operator *vec*. Operator *vec* takes the columns of the matrix in the argument and arranges them one by one in vertical direction into a vector. The relation between output matrix **V** and the vector form of the output can be expressed by

$$\mathbf{v} = vec\,(\mathbf{V}) = [v_1, v_2, \ldots, v_n]^T \tag{1}$$

$$v_i \geq 0 \text{ for all } i \in \langle 1; n \rangle$$

Using the state vector **v** the confinement criteria can be expressed in matrix form by

$$\mathbf{A}.\mathbf{v} = \mathbf{b} \tag{2}$$

In the case of the priority switching problem the dimension of matrix **A** will be $2n \times n^2$, where n is the number of the ports. The structure of the first n rows is as follows: the first row will start with a block of n ones continued with n-1 blocks of n zeros. The second row will start with a block of n zeros, continued with a block of n ones and n-2 blocks of n zeros. The third row will start with 2 blocks of n zeros, a block of n ones and n-3 blocks of n zeros, etc. Beginning with the $(n+1)^{th}$ row the structure will be different. In the $(n+1)^{th}$ row there will be ones at the 0^{th}, n^{th}, $2n^{th}$, etc. positions. In the $(n+2)^{th}$ row there will be ones at the 1^{st}, $(n+1)^{th}$, $(2n+1)^{th}$, etc. positions. All the other elements will be zero. This structure is shown in (3).

$$\mathbf{A} = \begin{pmatrix} 1 & 1 & .. & 1 & 0 & ... & & & & & 0 \\ 0 & ... & & 0 & 1 & 1 & ... & 1 & 0 & ... & 0 \\ . & & & & & & & & & & . \\ . & & & & & & & & & & . \\ . & & & & & & & & & & . \\ 0 & ... & & & & & ... & 0 & 1 & 1 & ... & 1 \\ 1 & 0 & ... & 0 & 1 & 0 & ... & 0 & 1 & 0 & ... & 0 \\ 0 & 1 & 0 & ... & 0 & 1 & 0 & ... & 0 & 1 & 0 & 0 \\ . & . & & & & . & & & & . & & . \\ . & & . & & & & . & & & & . & . \\ . & & & . & & & & .. & & & & . \\ 0 & ... & & 1 & 0 & ... & 0 & 1 & 0 & ... & 0 & 1 \end{pmatrix} \tag{3}$$

All elements in vector **b** will be ones.

$$\mathbf{b} = (1, 1, \ldots, 1)^{\mathrm{T}} \tag{4}$$

If equation (2) is valid, the first n rows of matrix **A** will guarantee that only one active element will be in each row of the output matrix. The following n rows will guarantee that only one active element will be in each column of the output matrix, and their combination into matrix **A** will guarantee that the total number of active elements will be n. In many cases the rows of matrix **A** will be linearly dependent. Since the determinant of matrix **A** will be calculated later, the number of rows of matrix **A** (and vector **b** correspondingly) must be reduced to make them linearly independent.

The object function of the optimization problem is used to evaluate the suitable solutions. Depending on the optimization problem the object function is either minimized or maximized. In our case the object function equals the sum of the priorities of the frames selected for transfer. During the solution of the priority switching problem we seek an object function with the lowest value. The object function $f(\mathbf{v})$ can be expressed as

$$f(\mathbf{v}) = \mathbf{c}^{\mathrm{T}}\mathbf{v} \tag{5}$$

where vector **c** contains weights assigned to the corresponding frames. Vector **c** is derived from the optimization matrix **C** using operation *vec*. If the frame is selected, the corresponding weight value is added to the total sum.

From the knowledge of matrix **A** (3) and vector **b** (4) the transformation function (6) can be constructed

$$\mathbf{v} \leftarrow T_{zs}.\mathbf{v} + \mathbf{s}\,. \tag{6}$$

Transformation matrix $\mathbf{T}_{\mathrm{zs}}$ is defined by (7) and vector **s** is defined by (8).

$$\mathbf{T}_{\mathrm{zs}} = \mathbf{I} - \mathbf{A}^{\mathrm{T}}(\mathbf{A}\mathbf{A}^{\mathrm{T}})^{-1}\mathbf{A} \tag{7}$$

$$\mathbf{s} = \mathbf{A}^{\mathrm{T}}(\mathbf{A}\mathbf{A}^{\mathrm{T}})^{-1}\,\mathbf{b} \tag{8}$$

The transformation matrix (6) ensures the convergence of the iteration process to a solution fulfilling the confinement criteria. Of course, fulfilling the confinement criteria is not enough. The solution also must minimize object function (5). This is achieved when the state vector **v** is updated during the iteration process by $d\mathbf{v}$, where

$$d\mathbf{v}/dt = \mathbf{T}_{\mathrm{op}}.\mathbf{v} + \mathbf{i}_{\mathrm{op}} \tag{9}$$

and $\mathbf{T}_{\mathrm{op}}$ and $\mathbf{i}_{\mathrm{op}}$ are expressed by (10) and (11)

$$\mathbf{T}_{\mathrm{op}} = \gamma\,(\mathbf{T}_{\mathrm{zs}} - \mathbf{I}) \tag{10}$$

$$\mathbf{i}_{\mathrm{op}} = \gamma\mathbf{s} - \mathbf{c} \tag{11}$$

More details about these transformations can be found in [2] and [3].

3 Neural network controlled switch fabric

A neural network controlled switch architecture is shown in Fig. 1. Such an architecture but with another type of neural network was presented in [4].

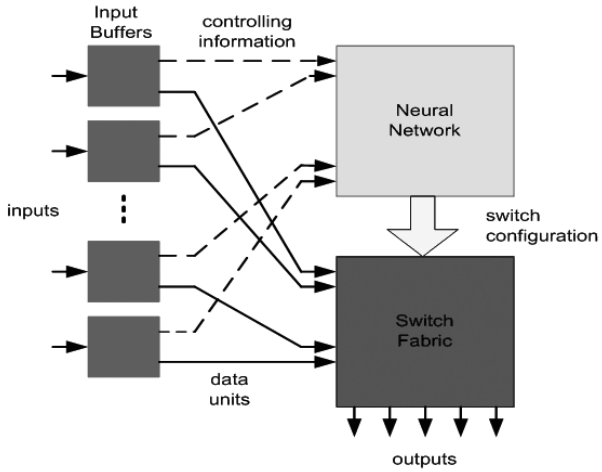

Figure 1: Neural network controlled frame switch

From the previous chapter it can be seen that the neural network realizes two separate operations. The first operation is related to the confinement criteria and the second is related to the object function. The neural network can be modelled by a block diagram shown in Fig. 2.

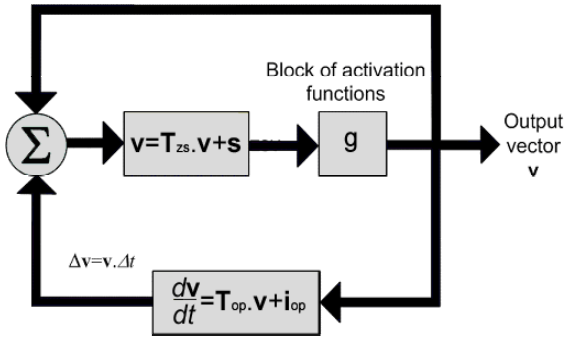

Figure 2: Block diagram of the neural network

The block diagram of the neural network consists of two loops. The upper loop is responsible for the confinement criteria and the lower loop is responsible for minimizing object function (5). The operations related to the confinement criteria can be separated into two parts. The first part, defined by (6), ensures that the sum of all elements of the state matrix **V** in each row and each column will be equal to 1. The second part, the neurons activation function

$$g(u_{xi}) = \begin{cases} 0 & \text{for} \quad u_{xi} < 0 \\ u_{xi} & \text{for} \quad 0 \le u_{xi} \le 1 \\ 1 & \text{for} \quad u_{xi} > 1 \end{cases} \tag{12}$$

ensures that the output values will remain within the range <0; 1>.

4 Analysis of priority levels

The aim of the analysis was to specify the impact of the amount of priority levels on the efficiency of the iteration process. There are two contradictory requirements on the amount of priority levels. First, the larger the amount of priority values the finer the division of the traffic processed by

the switch. From another point of view, increasing the number of priority levels increases the hardware requirements. It also substantially increases the number of required iteration steps. This leads to an increase in operation time, which is very critical in the case of fast frame switching.

As it came out from the results of the analysis, an extremely small amount of priority values messes up the algorithm and leads to invalid solutions. The amount of priority levels was determined by the number of bits used to express the elements of the optimization matrix **C**.

The Matlab environment was used to create the simulation model used for the analysis. The results generated by the Matlab model were evaluated from two points of view. First, the number of iteration steps needed to converge to a stable state value was evaluated. Second, the successfulness of the iteration process was tested. The iteration process was considered successful if the generated result fulfilled the criteria specified earlier.

4.1 Impact of the amount of priority levels

The next charts summarize the results of extended testing focused on the impact of the amount of priority levels. The analysis was performed for several numbers of ports n in the range <3; 15>. For each value of n, 14 different numbers of bits were tested in the range <1; 20>. With each number of bits, 40 independent tests were executed with randomly generated state and optimization matrixes.

The first chart in Fig. 3 shows the relation between the number of bits used to express priority levels and the average number of iterations needed to reach the stable state for n = 3, 9 and 15. Only successful iteration processes, i.e. those whose final stable state fulfilled the confinement criteria, were considered.

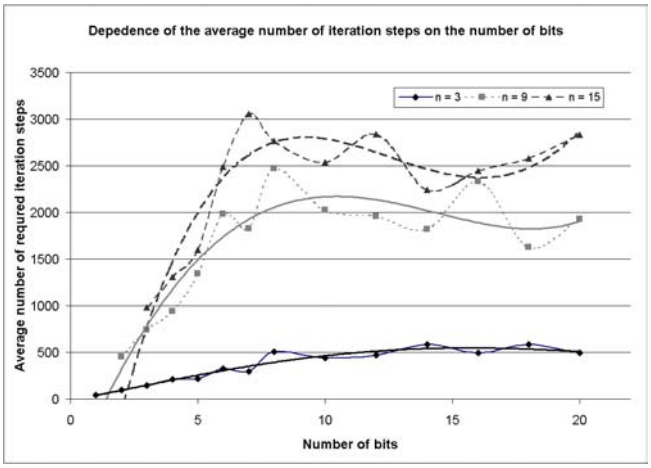

Figure 3: Dependence of the average number of iterations on the number of bits

The second chart in Fig. 4 shows the relation between the number of bits used to express priority levels and the successfulness of the iteration process for n = 3, 9, 15.

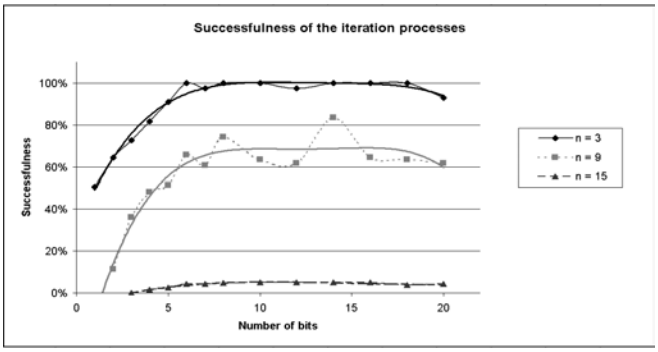

Figure 4: Dependence of the successfulness of iteration on the number of bits

The previous charts show that by increasing the number of bits up to a certain value (7 or 8, depending on n) the successfulness is rapidly increasing, but still further increase has no significant effect on the successfulness. It also can be seen that for very large values of bits (about 18 or 20) the successfulness starts to decrease. On the other hand, by increasing the number of bits the average number of iterations is increased. At the beginning this rise is quite steep, but for larger values it becomes more moderate.

5 The MATLAB environment created for the analysis

A MATLAB environment with graphical user interface (GUI) was developed for analysis of the priority switching problem, implementing the iteration process described in the preceding chapters. The graphical interface proposes the user to access the main parameters of the simulation process in a fast, convenient and intuitive way. Such parameters are number of nodes (i.e. ports), delta t, number of bits representing the elements of the matrix of conditions etc. The GUI allows the user to track the evolution of the iteration process in a graphical sub-window. In addition, the possibility to run a batch is implemented in the GUI. This can be very useful for needs of statistical analysis of a defined problem's configuration – e.g. for a given number of nodes.

5.1 Overall Decription of the GUI

The file containing the GUI is an *m-file* executable in the MATLAB environment (version 6.1 and later). The main window is shown in Fig. 5. The buttons managing different program functions are arranged at the right side of the window. The figure representing graphically (using the grey-scale) the degree of priority between all the nodes in the network is located in the upper-left part. The button 'Run' located under the figure runs the iteration process. The switch 'Step iteration' allows the user to switch to the step mode (see below). The text field in the bottom-left part is informing about the current number of iteration cycles. The most bottom-right part of the window is reserved for the switch, which enables/disables recording actions to the external file.

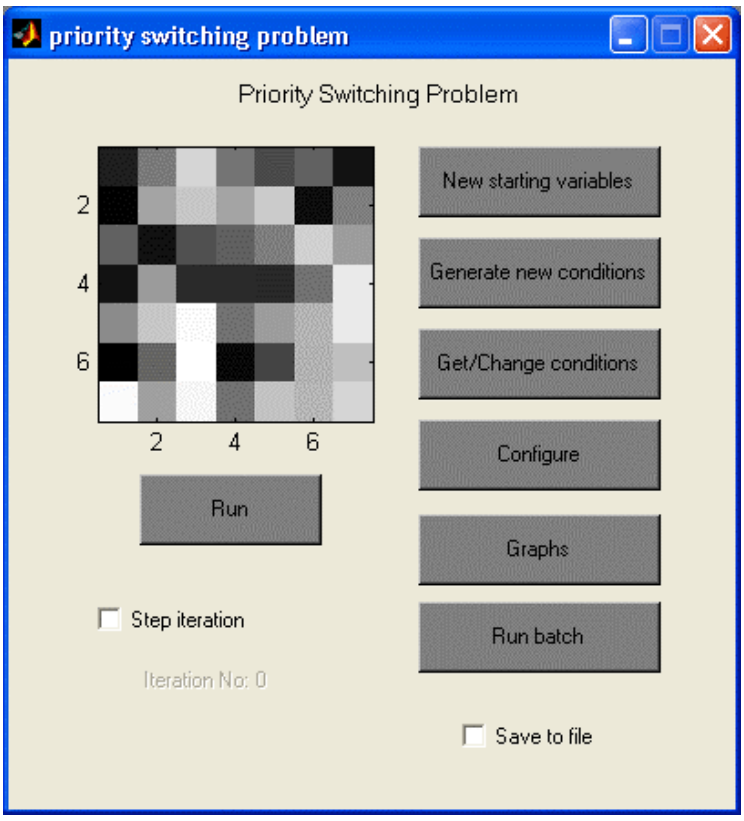

Figure 5: Main window of the GUI

During working with the GUI, additional information is displayed in the MATLAB Command Window, such as information about the course, about parameter changes etc.

5.2 Description and Functionality of the GUI Components

Next, the particular components of the GUI are described. The description also implies the way of their use.

Graph of the switching priorities sub-window. (see Fig. 6) The graph represents the switching priority between all the nodes in the network. In fact, the color boxes correspond to the elements of the matrix **V**. The elements of this matrix are real numbers between 0 and 1 included and each box in the graph is colored according to the value between black and white.

Each iteration process begins with random matrix **V** and this matrix is recomputed during the process. At the end of the process (if it is "successful", which is determined by the lower and upper threshold parameters, see below), all the elements of **V** are rounded so that it contains just black and white boxes (i.e. zeros and ones) as the result. In the 'step iteration' mode, the graph is redraw after each iteration step.

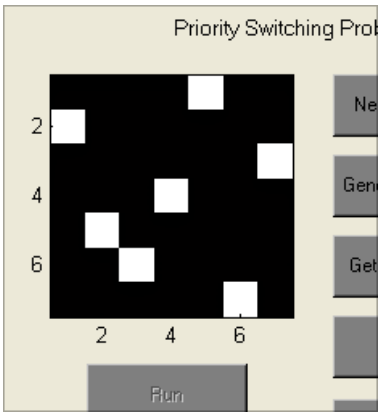

Figure 6: Sub-window representing the switching priorities (result of a successful process)

'Run' and 'Step' button. The button runs the iteration process. The process stops if one of the following conditions is satisfied:

- matrix **V** computed in the latest iteration cycle satisfies the conditions prescribed to be the final solution matrix (the *successful case*),
- the number of iteration cycles ('iteration No') reached the limit set in the Configure menu before **V** became the final matrix (the *unsuccessful case*).

After the process stops, the button gets disabled and it is no more possible to push it. In the successful case, you can read the number of iterations needed to reach the final solution in the text field, in the unsuccessful case the field turns red.

In the 'Step iteration' mode (see below), the 'Run' button changes to 'Step'. With this button you can now perform the iteration step-by-step, i.e. cycle per cycle. The current number of the iteration cycles is displayed in the text field. In the graph sub-window you can trace the development of the iteration process. In the 'Step iteration' mode the limit of the maximum number of iterations is disabled.

'Step iteration' checkbox. Checking this box turns the program into 'step' mode, i.e. you can trace the iteration process cycle-by-cycle (see above).

'New starting variables' button. Pushing this button generates new variables, which are used at the beginning of the process. Specifically, **v_old**, **A**, **b**, **Tzs**, **S**. The program is "reset" to the starting point (however, the matrix of conditions **C** remains the same!). The counter of number of iterations is also reset to zero.

'Generate new conditions' button. After pushing this button a new matrix of conditions, **C**, is generated. The elements of the matrix are independent random numbers with the uniform distribution

of probability between 0 and 1. Moreover, the bit precision of elements of the matrix **C**, set in the 'Configure' dialog, determines the values of these numbers: If the bit prescision is p, then there are exactly 2^p numbers spread between 0 and 1 to be candidates for the elements of **C**. For example, in case p=2, the matrix **C** can consist just from numbers 0, 0.3333, 0.6666, 1. Such generation is provided by the program sub-function.

'Get/Change conditions' button. The user can achieve two effects by pushing this button. If there is a matrix named 'C' in the MATLAB base workspace and this variable contains a square matrix with elements in the range 0 to 1, this matrix is loaded as the new matrix of conditions.

In the case there is no such matrix in the base workspace, the current matrix of conditions is just printed out in the Command Window.

'Configure' button. Pushing this buton opens a new dialog box – the configure dialog. In this dialog (see Fig. 7), the user can configure (set) all the important values that have effect on the iteration processes. Below is the description of each single parameter.

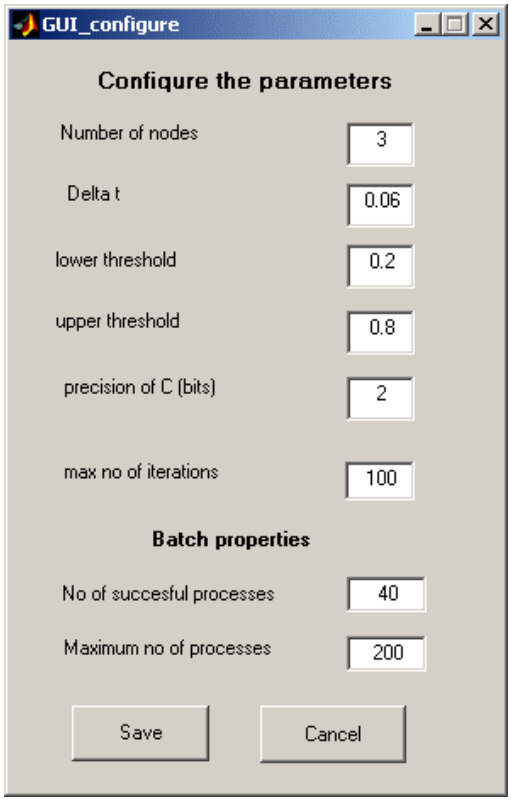

Figure 7: The Configure dialog

Parameters natively determining the iteration process:

A brief description of the parameters follows; a more detailed explanation of each of them can be found in the previous text and in [1,6].

Number of nodes	number of nodes in the switching problem (i.e. number of ports)
Delta t	variable determining the "duration" of the single step in each iteration cycle
lower threshold	parameter determining (along with the subsequent one, the upper threshold) if the matrix of priorities at the end of each iteration cycle is taken as the final result or not
upper threshold	analogic to the previous one
precision of C (bits)	number of bits which are used in generating the matrix of conditions, **C**

Other parameters:

max no of iterations how many iteration cycles are allowed to execute for each process

Batch parameters:

No of successful processes setting this parameter to *n* reflects in that the batch stops after reaching *n* successful processes

Maximum no of processes this parameter defines the total number of processes allowed to run within the batch (i.e. successful and unsuccessful together)

'Graphs' button. Opens new figure containing a graph related to the current iteration process. It is the 'Energy function plot', with energy defined by formula mentioned in the thesis [8]. The plot represents the dependency of the energy function on the number of iterations, i.e. the development of the energy throughout the iteration process. Examples of such graphs can be seen in Fig. 8.

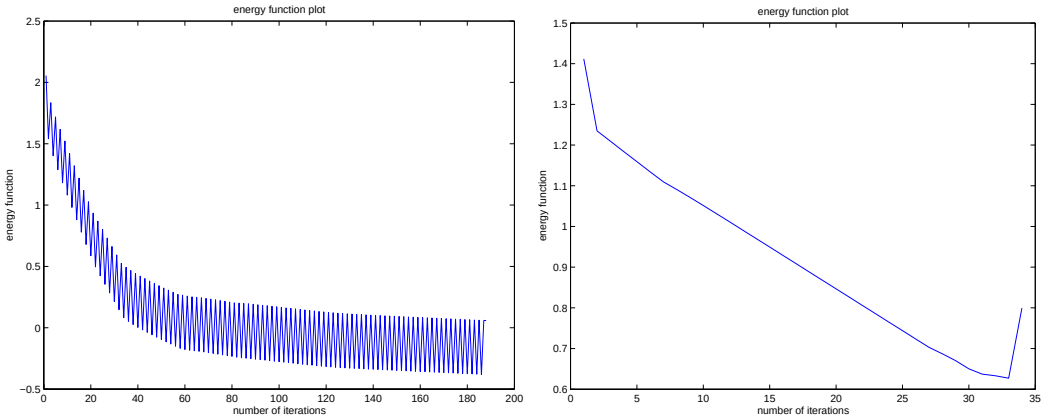

Figure 8: Examples of energy plots

'Run batch' button. This button starts a batch, i.e. a series of several iteration processes. This function is integrated to the GUI because it is very convenient for utilizing in larger analyses. In fact, this button simulates alternate pushing the 'Generate new conditions' and 'Run' buttons. In this way, it saves the user's time and effort.

Notice that the results of the batch processing are printed not just to the Command Window, but also to the output file, if it is open (see below, 'Save to file' checkbox).

'Save to file' checkbox. Checking this box will open the `report.csv` file in the MATLAB working directory (*and delete all its contents!*) to store the results of the iteration processes. If there is no file with this name, the program creates it. Unchecking the box stops the recording and closes the file.

The data are organized in the common CSV (Comma Separated Values) format. This is useful mainly for exporting the data into table processors like Microsoft Excel for further analyses. An example of the *report.csv* file is shown in Fig. 9.

6 Conclusion

In the paper, the model of priority switching problem and its mathematic apparatus has been presented. A graphical user interface for MATLAB was introduced which allows the user to virtually configure the switching fabric, to run experiments simulating the real situation and to get results from it.

The main focus has been on the investigation how the number of priority levels affects the speed of finding the optimal configuration of the switch in different conditions. Based on the results of the analysis some general statements about the behaviour of the system can be formulated: As the

number of nodes is increasing, the overall successfulness of processes is decreasing. Within the scope of a given number of nodes with 1 byte used to express the priority values the neural network operates effectively and the number of required iteration steps is relatively low. As the number of bits increases, the average number of iteration cycles increases. This trend culminates at about 10–14 bits and for more bits the average number of iteration cycles tends to decrease moderately.

If a single process does not reach the solution within the number of about four-five times the average number of iteration cycles, it is highly probable that the process will not converge to a solution at all.

```
     * FILE OPENED 15-Nov-2004 12:20:43
     ************************************
     1:,116,iterations needed
     * Generating new matrix of conditions C
     * Maximum of iterations (154) reached. The process has been stopped
prematurely.
     * Generating new matrix of conditions C
     * Maximum of iterations (154) reached. The process has been stopped
prematurely.
     * Generating new matrix of conditions C
     2:,168,iterations needed
     3:,166,iterations needed
     4:,228,iterations needed
     5:,178,iterations needed
     6:,144,iterations needed
     7:,160,iterations needed
     * Generating new matrix of conditions C
     8:,313,iterations needed
     * Generating new matrix of conditions C
     * Maximum of iterations (15400) reached. The process has been stopped
prematurely.
     * Generating new matrix of conditions C
     * Maximum of iterations (15400) reached. The process has been stopped
prematurely.
     * Generating new matrix of conditions C
     * Maximum of iterations (15400) reached. The process has been stopped
prematurely.
     * Generating new matrix of conditions C
     * Maximum of iterations (15000) reached. The process has been stopped
prematurely.
     ************************************
     FILE CLOSED 15-Nov-2004 16:17:36
```

Fig. 9: An example of the report.csv file

References

[1] K. Molnár, V. Vrba, Frame priorization support for switch fabrics, *3rd International Conference on Networking*, Gosier, 2004, pp. 141 - 146, ISBN 0-86341-326-9

[2] S.V. Balakrishnan-Aiyer, *Solving combinatorial optimization problems using neural networks with applications in speech recognition*, University of Cambridge, United Kingdom, 1991

[3] A.H. Gee, *Problem solving with optimization networks*, University of Cambridge, United Kingdom, 1993

[4] K.A. Smith, *Solving combinatorial optimization problems using neural networks*, University of Melbourne, Australia, 1996

[5] T. Troudet, S.M. Walters, Neural Network Architecture for Crossbar Switch Control, *IEEE Transaction on Circuits and Systems*, vol. 38, No. 1, 1991

[6] K. Molnár, P. Rajmic: Impact of priority levels on the efficiency of priority switching, Proceedings of the 4th WSEAS International Conference on: Telecommunications and Informatics (TELE-INFO'05), ISBN: 960-8457-11-4, pp. 48-51, Prague, 2005
[7] The MathWorks: *MATLAB - Creating Graphical User Interfaces (Version 6)*. PDF Help to MATLAB release 13. 2002.
[8] K. Molnár, Aplikace umělých neuronových sítí ve vysokorychlostních aktivních síťových prvcích (Application of artificial neural networks in high-speed active network elements), in Czech, Ph.D. Thesis, Brno University of Technology, 2002.

This paper has been supported by the Grant Agency of the Czech Republic (Grants No. 102/03/0560, 102/05/P585, 102/03/0762) and the Ministry of Education of the Czech Republic (project No. 1K04116).

Pavel Rajmic, Karol Molnár
Department of Telecommunications, Faculty of Electrical Engineering and Communication
Brno University of Technology
Purkyňova 118, 618 00 Brno, Czech Republic
rajmic@feec.vutbr.cz, molnar@feec.vutbr.cz
phone: +420 541 149 166