The method of optimization of neuro-based concurrent operations in neurocomputers

V A Romanchuk

Ryazan State University named for S Yesenin, 46, Svobody ave., Ryazan, 390000, Russia

E-mail: v.a.romanchuk@rsu.edu.ru

Abstract. The article deals with the task of optimization of neuro-based concurrent operations to be implemented in neurocomputers. We define mathematical tools of this optimization that can employ the set-theoretic approach towards such concepts as task, operation, and microcommand. We consider segmentation and parallelization of operations as methods to use, depending on precedence relations among operations that constitute these segments. The task solution of optimization of neuro-based concurrent operations in neurocomputers can be applied to a whole class of neurocomputers, regardless of the manufacturer, the model or the product line, since we only address the general properties and principles of the neurocomputer operation. We select criteria and define methods of evaluating the effectiveness of parallelization of concurrent operations, when they are implemented in neurocomputers. We describe our empiric research in the form of a software system that automatically optimizes neuro-based concurrent operations in neurocomputers on the NP Studio platform.

1. Introduction

For a long time, neurocomputers were considered effective for solving so-called non-formalized and low-formalized tasks in situations when this requires an algorithm of task-solving training with authentic empiric data. The first type of these tasks was connected with approximation of specific functions to a discreet set of values, i.e., the task of image recognition [1]. At present, neurocomputers are widely used in designing data processing systems and in control of complex objects (including manufacture of machinery).

By the present time, another class of tasks has been added, those which do not always require training and empiric data, but are easy to represent in a neuro-network logic basis. Among them, we see tasks that display expressed natural parallelism in processing of signals, images, et al. We find the proof of better prospects for neurocomputers than for other architectural solutions in the fact that recently, the range of general mathematical tasks solved on a neuro-computer basis has increased. We can also list such tasks as solving large-size linear and non-linear algebraic equations and inequations; systems of non-linear differential equations; partial-derivative equations; optimization procedures, and other tasks [2, 3].

Yet currently, there is an issue with the lack of mathematical tools and optimization techniques for neuro-networking tasks for neurocomputer solutions, which renders this task solving ineffective. The results of this research have been tested during development of algorithms and software for a specialized computing device based on neuroprocessors, whose function is to automatically control...
electromechanic system modules (illustrated in a hexapode device) in a near-real time mode.

2. Mathematical tools of optimization of neuro-based concurrent operations in neurocomputers

Let us suppose that $Z^{(j)}$ is a certain neuro-networking task in a neuroprocessor platform implementation. Neuro-network task $Z^{(j)}$ can be set as a tuple of parameters and specifications:

1. A set of artificial neuron network (ANN) inputs $X = \{x_1, ..., x_i, ..., x_{na}\}$, where each $x_i$ input is defined by its digit capacity $X_{Ri}$ and by the $X_{Ni}$ type.

2. A set of artificial neuron network inputs $Y = \{y_1, ..., y_j, ..., y_ya\}$, where each $y_j$ output is defined by its digit capacity $Y_{Rj}$ and by $Y_{Nj}$ types.

3. A set of neurons $N = \{n_1, ..., n_j, ..., n_{na}\}$, where each requires emulation for solving the $Z^{(j)}$ task.

4. A set of neuron associations defines the dependencies among the artificial network neurons: $N_{E} = \{N_{11}, N_{12}, ..., N_{inc}, ..., N_{nac}\}$, which can be represented as a connection matrix of neurons, inputs and outputs of the artificial neuron network.

$N_{E} \rightarrow MNet = [MNet_{ij}]$ with dimensions $(nc + xn + yn) \times (nc + xn + yn)$.

This matrix shows an artificial neuron network topology, the number of ANN layers, availability or the absence of bypass connections, transfer functions of neurons.

We introduce the notion of precedence, $\prec$, set as $\forall n_i, n_j \in N, n_i \prec n_j$, where, for its input, the $n_j$ neuron employs the output of the $n_i$ neuron, that is why, the emulation of the $n_j$ neuron can run without preliminary emulation of the $n_i$ neuron.

Each element of the matrix can take these values: $MNet_{ij} = \{\sim, \sim', \sim''\}:

'\sim'$ (no) – no precedence relationship $\sim (n_i \prec n_j)$ between subprogrammes $n_i$ and $n_j$ ($j > i)$;

'\sim'' (yes) - precedence relationship $n_i \prec n_j$ ($j > i$) between subprogrammes $n_i$ and $n_j$;

'\sim'$ - precedence relationship is impossible between neurons.

As an illustration, we can consider the matrix of associations between six neurons $N = \{n_1, n_2, n_3, n_4\}$, two inputs $X = \{x_1, x_2\}$ and single output $Y = \{y_1\}$, the matrix having a perceptron topology (Figure 1).

![Figure 1](image-url) - The illustration of a perceptron to set a matrix of associations.
In the $MNet$ matrix, all elements below the main diagonal line equal 'n', as a perceptron does not have back links.

To solve the $Z^{(j)}$ task, all neurons require emulation in accordance with the relationships of precedence between neurons; then the $Z^{(j)}$ task can be represented as a tuple of neuron emulation operations like this: $y = f(g) = f\left(\sum_{i=1}^{n} a_i x_i + a_0\right)$.

We introduce the set of operations $O = \{O_1, O_2, ..., O_{nc}\}$. Then, neuro-network task $Z^{(j)}$ is a tuple of operation $O_1, O_2, ..., O_m, ..., O_{nc}$ with a length of $nc = |Z^{(j)}|; j = 1, N$, i.e.: $Z^{(j)} = <O_1, O_2, ..., O_m, ..., O_{nc}>; j = 1, N$.

Thus, the number of operations $O_1, O_2, ..., O_m, ..., O_{nc}$ corresponds to the number of emulated neurons of the ANN, as a certain $O_i$ operation represents a mathematical model of a formal neuron: $O_i = f\left(\sum_{m=1}^{n} a_m x_m + a_0\right)$.

For certain operations $O_i$ and $O_j$, the precedence relationship $<$ is also applicable, and it is set in this way: $\forall O_i, O_j \in O, O_i < O_j$ : for input, the $O_j$ operation employs the output data of the $O_i$ operation, so the $O_j$ operation cannot run unless the $O_i$ operation has been performed.

The $O_m$ operation can be regarded as a set of autonomous operations, based on the associations and functional units of a neurocomputer that implement the set: $O_m = \{O_{m,1}, O_{m,2}, ..., O_{m,NP}\}$, where $NP$ is the number of possible functional stages of neuron emulation operations in the neuroprocessor.

In general, the number of operations during the neuron emulation is as follows: $NP = xn + (xn - 1) + 1$, where $xn$ is the number of neuron inputs, depending on the ANN.

Here, the first term defines the $u_m$ operation that multiplies the $x_m$ input value by the $a_m$ link weight. The second term defines the $s_m$ operation of adding the results of the multiplication product. The third term is the operation of activating the $a_m$ neuron.

Precedence relationship for specific functional modules of a single neuronal emulation operation can be represented as an $MNeuron$ matrix with dimensions $(2 \times xn + (xn - 1) + 2) \times (2 \times xn + (xn - 1) + 2)$. Then, the ANN represented in Figure 1 can be represented as a set of concurrent operations (Figure 2).
Figure 2. The illustration of a perceptron represented as a tuple of operations.

Then, for a specific $Z^{(j)}$ task, the $MNetA$ matrix, describing all precedence relationships between operations can be defined as a combination of $MNeuron$ and $MNet$ matrices, where each element of the $MNet_{ij}$ matrix can be defined as an $\gamma$ image, i.e., as an $MNeuron$ matrix:

$$\gamma : MNet_{ij} \rightarrow MNeuron.$$  

The precedence relationships of the $MNet$ matrix are preserved as those associated with input and output values of the $MNeuron$ matrix. The dimensions of the $MNetA$ matrix are as follows:

$$(nc + xn + yn) \times (2 * xn + (xn - 1) + 2) \times (nc + xn + yn) \times (2 * xn + (xn - 1) + 2).$$

At the stage of software development, each operation $O_i$ or set of operations $\{O_{m,1}, O_{m,2}, ..., O_{m,NP}\}$ is assigned a tuple of neuroprocessor microcommands in its internal coding $<MK_1^i, MK_2^i, ..., MK_K^i>$; where $K$ is the minimum number of microcommands necessary for implementation of a certain $O_i$ operation. Most commonly, since a neurocomputer is a highly concurrent device, a single microcommand may correspond to a tuple of operations of neuron emulation: during a single cycle of a neuroprocessor, a set of neurons can be emulated. Then, several variants are possible:

- $O_i \rightarrow MK_1^i, MK_2^i, ..., MK_K^i ; K > 0$
- $O_i \rightarrow MK_1^i, MK_2^i, ..., MK_K^i ; K = 1$
- $O_i \rightarrow MK_1^i, MK_2^i, ..., MK_K^i ; K = 1$

The task of optimization consists in defining operation tuples that correspond to a microcommand, in order to maximize the number of neuron emulation operations in a specific microcommand:

$$\forall MK_p \in MK : MK_p \rightarrow O_1, O_2, ..., O_l, ..., O_K ; R \rightarrow \max.$$  

Neurocomputers can be divided into two groups, depending on the neuron emulation methods [4]:

1. Neurocomputers that preform neuron emulation microoperations at the hardware level without assigning specific functional units to specific concurrent operations (addition, multiplication, activation).

2. Neurocomputers that perform neuron emulation microoperation at the hardware level stage after the stage, assigning specific functional units to specific concurrent operations (addition, multiplication, activation).

In the first case, since it is impossible to assign special functional units, we employ the $MNet$...
matrix as a method of data input. In the second case, it is more effective to use the \( MNetA \) matrix, as this yields better response time, due to reassignment of tuples of concurrent operations.

The goal of the stage of reassignment of concurrent operations \( Z^{(j)} = \langle O_{1}, O_{2}, ..., O_{m}, ..., O_{nc} \rangle; j = 1, N \) and their rearrangement in \( p \) operation tuples is:

\[
\langle O_{1}, O_{2}, ..., O_{nc} \rangle \rightarrow \langle O_{1}, O_{12}, ..., O_{unc} \rangle, ..., \langle O_{p1}, O_{p2}, ..., O_{pm} \rangle
\]

for their further representation in neuron emulation microcommand(s): \( \langle O_{11}, O_{12}, ..., O_{unc} \rangle, ..., \langle O_{p1}, O_{p2}, ..., O_{pm} \rangle \rightarrow \langle MK_{1}, MK_{2}, ..., MK_{K} \rangle \).

We introduce the notion of a segment of operations, \( SO \), i.e., a tuple of \( O \) operations, without precedence relationships between them: \( SO_{i} = \langle O_{i1}, O_{i2}, ..., O_{unc} \rangle; O_{i} \rightarrow \langle O_{i1}, \ldots, O_{inc} \rangle \).

Then the whole tuple of operations that implement the \( Z^{(j)} \) task can be divided into a tuple of segments: \( \langle O_{1}, O_{2}, ..., O_{m}, ..., O_{nc} \rangle \rightarrow \langle SO_{1}, SO_{2}, ..., SP_{p} \rangle \), where \( SO_{i} < SO_{2} < \ldots < SP_{p} \).

Segmentation consists of the following stages:

1. Selection of a matrix of precedence relationships, depending on the neuroprocessor type. In the matrix for solving this task, we consider only the area of the neuroprocessor’s functional units, restricted to the precedence relationships of input and output data.

2. For each value in column \( MNet_{i} = \{n_{1}, n_{2}, ..., n_{m} \} \) of the selected area of the \( MNet (MNetA) \) matrix, the following conditions are set:
   - If for \( n_{i} \), all the values in a column equal 'n', the \( O_{i} \) operation (or a tuple of operations), corresponding to the \( n_{i} \) neuron emulation, can be assigned to any \( SO_{j} \) segment.
   - All \( n_{i}, i = 1, nc \) having the same distribution of 'y' values in matrix areas of the selected area are assigned to a specific \( SO_{j} \) segment.
   - If a back connection is necessary for solving a task, and if this connection has to be a precedence relationship between the last and the first neurons in the ANN, the 'y' value in the left-hand bottom corner of matrix \( MNet_{i} = \{n_{1}, n_{2}, ..., n_{nc} \} \) will not be considered.

3. With a set of segments \( \{SO_{1}, SO_{2}, ..., SP_{p} \} \), the precedence relationships can be defined in the following way:
   - if the lowest value 'y' of segment operations \( SO_{i} \) has a position in a matrix which is higher than the position of the lowest value 'y' in the operations of segment \( SO_{j} \), then \( SO_{j} < SO_{i} \).

With a tuple of segments that are connected by precedence relationships, in order to program and represent operations as a microcommand set of the processor, it is necessary to divide the segments, in order to select a tuple of operations performed in a single cycle, in the following way:

- if the number of operations \( |SO_{i}| \) in segment \( SO_{i} \) is higher than \( MAC \), it is necessary to split the segment in two, where precedence relationships between the segments will not be noted: \( |SO_{i}| > MAC : SO_{i} = SO_{i1} \cup SO_{i2} \).

As a result, we have a tuple of segments, each of them is represented as a microcommand (or a tuple of microcommands) with a maximum concurrence level of performing the operation set within a cycle.

Further optimization will be possible with semantic analysis of the \( Z^{(j)} \) task. Then it is possible to speak about effective parallelization of operations, which will be expressed through \( K_{MAC} \) and \( K_{MACi} \) in each processor cycle.

For the first type of neurocomputers, the effectiveness of parallelization can be calculated in the
following way: $K_{MAC,i} = \frac{M_i}{MAC}$, where $M$ is the number of neurons emulated on a neurocomputer within a cycle, and $MAC$ is the maximum number of neurons emulated on the specific neurocomputer within its cycle.

For the second type of neurocomputers: $K_{MAC,i} = \frac{M_{Oi}}{MAC * NP}$, where $M_{Oi}$ is the number of microoperations performed (addition, multiplication, activation) within a cycle, $NP$ is the number of microoperations that constitute neuron emulation operation $\{O_{m,1}, O_{m,2}, \ldots, O_{m,NP}\}$.

3. Empirical research

For convenience of processing various inputs, we have developed an application that automatically optimizes neuro-based concurrent operations in neurocomputers in the offered subsystem, 'Analysis and optimizations of neurocomputers' within our software system 'NP Studio' [6-10].

4. Conclusion

The theoretic and empirical research that suggest methods of solving operations optimization, when these are performed on neurocomputers, enable us to increase the speed, due to employing all the functional units of a neurocomputer for several classes of tasks, easily represented in a neuro-network logic basis. These are tasks that show pronounced natural parallelism in signal processing, image processing, et al.

References

[1] Pande S, Morgan F, Cawley S, Bruintjes T, Smit G, McGinley D, Carrillo S, Harkin J and McDaid L 2013 Neural Process Letters 38 131-153
[2] Izeboudjen N, Larbes C and Farah A 2014 Artificial Intelligence Review 41 491-534
[3] Kolinummo P, PASI Pulkkinen P, Hämäläinen T and Saarinen J 2000 Neural Processing Letters 12 171-82
[4] Ruchkin V, Romanchuk V and Sulitsa R 2013 Proc. of the 2nd Mediterranean Conf. on Embedded Computing (Montenegro: Budva) 58-62
[5] Ruchkin V, Fulin V, Kostrov B, Romanchuk V and Ruchkina E R 2015 Proc. of the 4nd Mediterranean Conf. on Embedded Computing (Montenegro: Budva) 45-50
[6] Avetisyan A, Grushin D, Ryzhov A 2002 Proc. of the Institute for System Programming of Russian Academy of Sciences 140-147
[7] Ivutin A, Larkin E 2014 Proc. of the 3rd Mediterranean Conf. on Embedded Computing (Montenegro: Budva) 236-239
[8] Diken E, Jordans R, Corvino R, Jozwiak L, Corporaal H and Chies F 2014 Microprocessors and Microsystems 38 947-959
[9] Jordans R, Jozwiak L, Corporaal H 2014 Proc. of the 3rd Mediterranean Conf. on Embedded Computing (Montenegro: Budva) 32-35
[10] Bekhtin Y, Gurov V, Guryeva M 2014 Proc. of the 3rd Mediterranean Conf. on Embedded Computing (Montenegro: Budva) 252-255