engineering proceedings

MDPI

Proceeding Paper

Artificial Neural Networks Multicriteria Training Based on Graphics Processors †

Vladimir A. Serov *, Evgenia L. Dolgacheva, Elizaveta Y. Kosyuk, Daria L. Popova, Pavel P. Rogalev and Anastasia V. Tararina

Department of Applied Information Technologie, MIREA—Russian Technological University (RTU MIREA), Moscow 119454, Russia; evgeniadolgacheva@yandex.ru (E.L.D.); kosyuk.ey@mail.ru (E.Y.K.); pdl13@yandex.ru (D.L.P.); radugapp@mail.ru (P.P.R.); nharvard2013@gmail.com (A.V.T.)

* Correspondence: ser_off@inbox.ru

† Presented at the 15th International Conference "Intelligent Systems" (INTELS'22), Moscow, Russia, 14–16 December 2022.

Abstract: The report considers the task of training a multilayer perceptron, formulated as a problem of multiobjective optimization under uncertainty. To solve this problem, the principle of vector minimax was used. A parallel software implementation of a hierarchical evolutionary algorithm for solving a multicriteria optimization problem under uncertainty based on a GPU is presented.

Keywords: artificial neural network; multicriteria optimization under uncertainty; vector minimax; parallel computing; GPU

check for updates

Citation: Serov, V.A.; Dolgacheva, E.L.; Kosyuk, E.Y.; Popova, D.L.; Rogalev, P.P.; Tararina, A.V. Artificial Neural Networks Multicriteria Training Based on Graphics Processors . *Eng. Proc.* **2023**, *33*, 57. https://doi.org/10.3390/engproc2023033057

Academic Editors: Askhat Diveev, Ivan Zelinka, Arutun Avetisyan and Alexander Ilin

Published: 25 July 2023

Copyright: © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Currently, the technology of the neuroevolutionary synthesis of management and decision-making models is being intensively developed, which is considered as a promising means of implementing intelligent algorithms for analyzing information and management under conflict and uncertainty in real time [1–5]. The effectiveness of the neuroevolutionary approach for solving this class of problems is determined by the ability to take into account uncertain factors, such as conflict uncertainty, the multicriteria of management goals, and the uncertainty of environmental conditions. In this context, it is expedient to formalize the task of training an artificial neural network (ANN) in the form of a multicriteria optimization problem under uncertainty (MCOU). In [6,7], a coevolutionary technology for solving the MCOU problem was developed, which, as the results of computational experiments show, has an extremely high computational complexity. In [8–12], it was shown that a promising area of research is the parallel implementation of computing technology based on graphics processors (GPUs). In this article, a parallel GPU implementation of a coevolutionary technology for solving the MCOU problem is proposed.

In Section 2, the formulation of the ANN training problem is formulated in the form of an MCOU problem, where the principle of vector minimax is applied for its solution. Section 3 presents a parallel GPU-based implementation of the MCOU hierarchical evolutionary algorithm software. Section 4 presents the results of a computational experiment on a test problem.

2. Problem Statement

The statement of the training problem for a multilayer perceptron (MP) is formulated as an MCOU problem:

$$\langle \mathbf{W,Z,F(w,z)} \rangle, \quad (1)$$

where $\mathbf{w} \in \mathbf{W} \subset \mathbf{E}^{r_w}$ is a vector of weight coefficients of MP synaptic connections; $\mathbf{z} \in \mathbf{Z} \subset \mathbf{E}^{r_z}$ is a vector of uncertain factors; $\mathbf{Z}$ is a finite set of possible values of the uncertain

factor; and $\mathbf{F(w,z)} = [f_1\mathbf{(w,z)}, \ldots f_m\mathbf{(w,z)}]^T \in \mathbf{E}^m$ is a vector criterion defined on the Cartesian product $\mathbf{W} \times \mathbf{Z}$.

In problem (1), it is required to determine the value of the vector $\mathbf{w} \in \mathbf{W}$, which provides the minimum values for the components of the vector criterion **F(w,z)** under the influence of an uncertain factor $\mathbf{z} \in \mathbf{Z}$, about which it is known only that it can take values from a finite set **Z**.

To solve this problem, it is proposed to use the vector minimax principle. In this case, the original statement of problem (1) is reduced to a deterministic multiobjective optimization problem:

$$\mathbf{V(w)} \rightarrow \min_{\mathbf{w} \in \mathbf{W}}, \tag{2}$$

where **V(w)** is a vector indicator, the components of which are the points of extreme pessimism of the vector criterion **F(w,z)** on the set **Z** with fixed **w**.

To solve problem (1), (2), the hierarchical evolutionary algorithm (HEA) of the MCOU developed in [6,7] is used. As studies [5,6] show, the HEA MCOU, when used in ANN training tasks, shows a high computational complexity. Therefore, it is proposed to implement the HEA software for solving problem (1), (2), based on the GPU architecture and OpenCL technology.

3. GPU-Based Parallel Implementation of the Hierarchical Evolutionary MCOU Algorithm

The architecture of the developed HEA MCOU software reflects the following main stages of the HEA MCOU implementation.

Stage 1. Formation of a set of points of extreme pessimism (Figure 1).

Step 1. The initial population **W**, $|\mathbf{W}| = n$ is formed on the host (CPU).

Step 2. Constant memory is allocated on the GPU, into which arrays **Z** and **W** are entered. A buffer is allocated in the global memory of the GPU for the set of points of extreme pessimism **V(W)**.

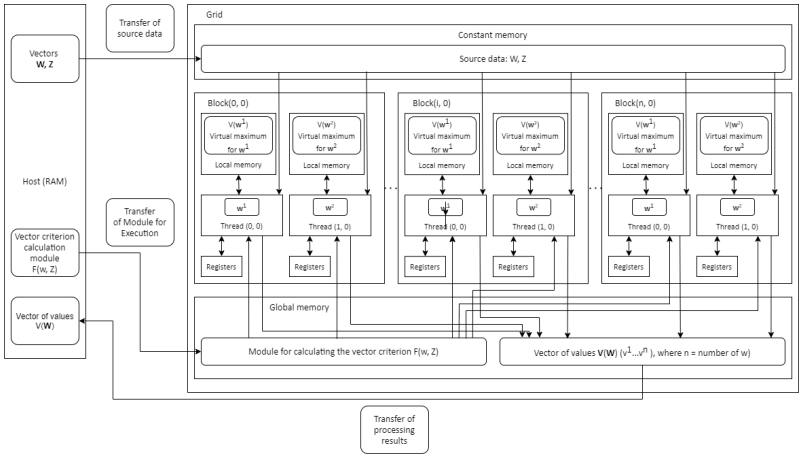

Figure 1. Parallel algorithm for finding a set of points of extreme pessimism on the GPU.

Step 3. A grid is formed on the GPU that determines the number of working blocks and the threads executed in them.

Step 4. The kernel is called with a subsequent transfer from the CPU to each thread of a set of instructions for execution. Threads start to work in parallel. Within each thread, the corresponding set of values of the vector criterion **F(w,Z)** is calculated for each $\mathbf{w} \in \mathbf{W}$ and the extreme pessimism point **V(w)** is calculated on the set **F(w,Z)**, which is stored in the local memory of the thread. Upon completion, each thread transfers its value **V(w)** to the global memory of the GPU. After the GPU has signaled that all threads have terminated, the CPU moves the array **V(W)** from the GPU's global memory to the CPU's RAM.

Stage 2. Assessment of the fitness of each point $\mathbf{w} \in \mathbf{W}$ (Figure 2).

Step 5. Constant memory is allocated on the GPU, into which array **V(W)** is entered. In the GPU's global memory, a buffer is allocated for the set of values of the fitness function $\Phi(\mathbf{V(W)})$.

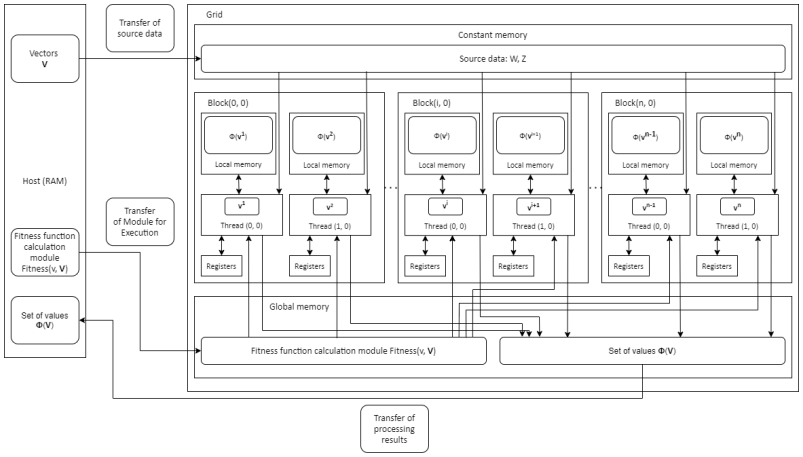

Figure 2. Parallel algorithm for calculating fitness function values on GPU.

Step 6. A new grid is formed on the GPU.

Step 7. The kernel is called and threads are started to work in parallel. Within each thread, the corresponding value of the fitness function Φ**(V)** is calculated for each element of $\mathbf{V} \in$ **V(W)**, which is stored in the local memory of the thread. Upon completion, each thread transfers its Φ**(V)** value to the GPU's global memory. After the GPU sends a signal to terminate all threads, the CPU transfers array Φ**(V(W))** from the GPU's global memory to the CPU's RAM.

Next, a population of descendants is formed on the CPU and the execution of stages 1, 2 is repeated.

The developed algorithm can be easily modified to solve the MCOU problem (1), (2), where there are many uncertain factors.

The developed software is cross-platform, as CUDA and OpenCL technologies are available on various operating systems, both on Windows and Linux.

4. Computational Experiment

The effectiveness of the developed technology was tested on the following test task MCOU:

$$\Gamma = \langle \mathbf{X, Z, F(x,z)} \rangle, \tag{3}$$

where $\mathbf{x} = [x_1, x_2]^T \in \mathbf{X}$ is the vector of control parameters; $\mathbf{z} = [z_1, z_2]^T \in \mathbf{Z}$ is the vector of uncertain factors; and $\mathbf{F(x,z)} = [f_1\mathbf{(x,z)}, f_2\mathbf{(x,z)}]^T$ is the vector performance indicator with components:

$$f_1\mathbf{(x,z)} = x_1^2 + x_2^2 - x_1(z_1^2 - z_2^2), \tag{4}$$

$$f_2\mathbf{(x,z)} = x_1^2 - x_2^2 - x_1(z_1^2 + z_2^2). \tag{5}$$

The restrictions were set in the form:

$$\mathbf{X} = \{0 \le x_1, x_2 \le 2\}, \tag{6}$$

$$\mathbf{Z} = \{\mathbf{z}^i, i = \overline{1, |\mathbf{Z}|} \; 0 \le z_1, z_2 \le 2\}. \tag{7}$$

It is required to maximize the components of the vector efficiency indicator on the set $\mathbf{X} \times \mathbf{Z}$ based on the vector maximin principle.

Figures 3–5 show the results of searching for a set of vector maximins using the HEA MCOU (elite points in each generation are highlighted in red). Algorithm parameters: population cardinality $|\tilde{\mathbf{X}}| = 1000$; $|\mathbf{Z}| = 1000$; real coding and SBX-crossover were used.

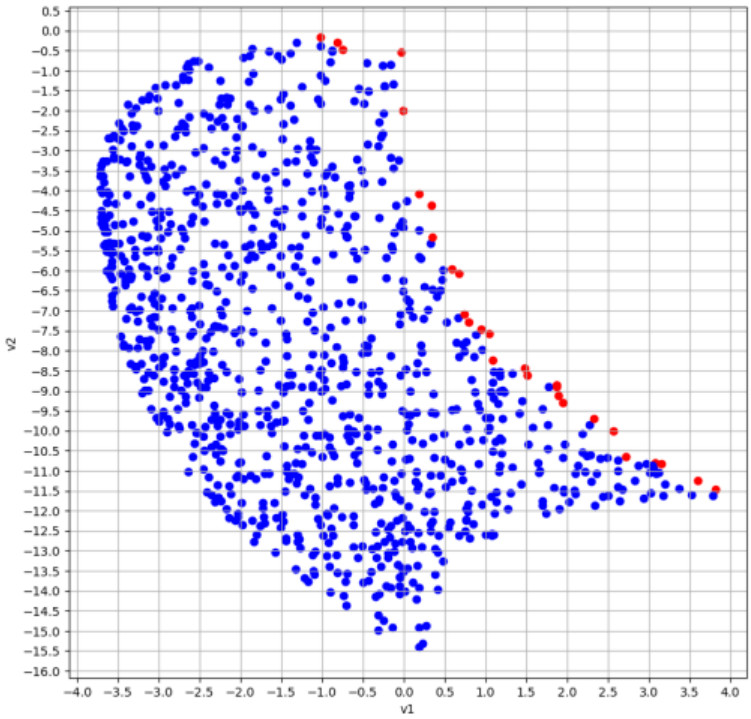

Figure 3. Evolutionary MCOU algorithm, generation No. 1. Elite points in each generation are highlighted in red.

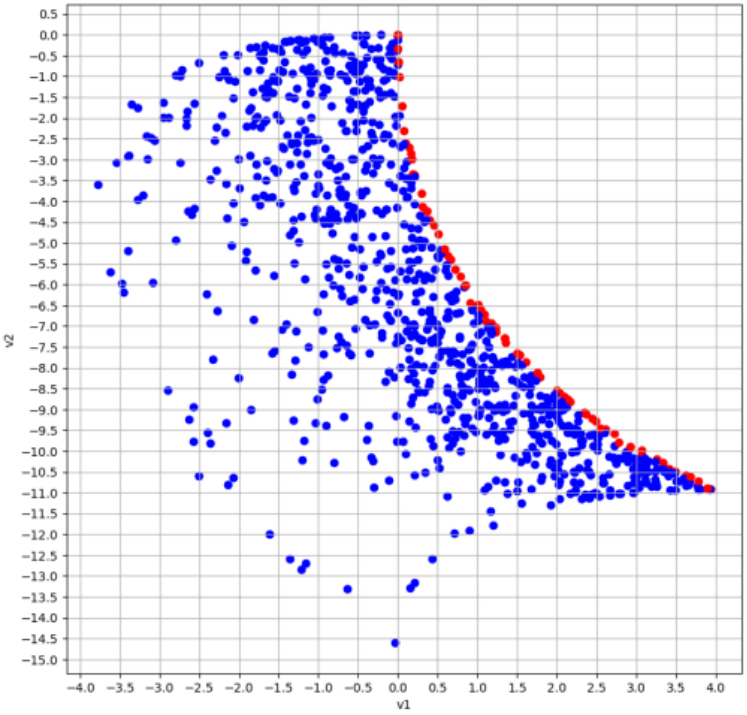

Figure 4. Evolutionary MCOU algorithm: generation No. 5.

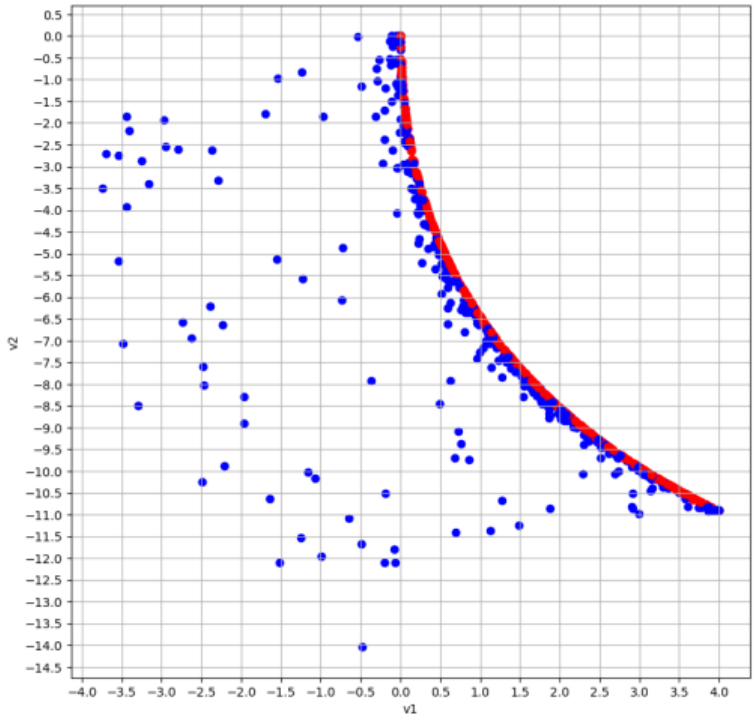

Figure 5. Evolutionary MCOU algorithm: generation No. 10.

Table 1 provides a comparative analysis of the running time of sequential and parallel evolutionary algorithms for solving the considered MCOU test task.

Table 1. Comparative analysis of sequential and parallel MCOU algorithms.

Population Size $\lvert\tilde{\mathbf{X}}\rvert$	Running Time of the Parallel Algorithm t_{par}, s	Running Time of the Sequential Algorithm t_{seq}, s
10	0.1353302	0.0003806
50	0.1354179	0.0049311
100	0.1397946	0.0129666
500	0.1476841	0.2834744
1000	0.1611041	1.0414738
5000	0.2373939	26.2836442
10,000	0.3495695	104.6257208
50,000	0.3617121	2623.5825713
100,000	0.6700136	10,134.2378412

A comparative analysis shows that, with a small population size $|\tilde{\mathbf{X}}| \leq 500$, the running time of the parallel evolutionary algorithm MCOU t_{par} is greater than or comparable to the running time of the sequential algorithm t_{seq}. This is due to the fact that the parallel algorithm spends additional time preparing and transferring data to the GPU. However, with a further increase in the size of populations, the advantage of the parallel evolutionary MCOU algorithm in relation to the sequential analog increases. In particular, for $|\tilde{\mathbf{X}}| = 100,000$, the running time of the parallel evolutionary MCOU algorithm is $t_{par} \cong 10^{-4} t_{seq}$.

5. Conclusions

The formulation of the MP training problem was formalized as an MCOU problem, where the vector minimax principle was used for its solution.

A parallel implementation of a hierarchical evolutionary algorithm for searching for a set of vector minimaxes in the MCOU problem based on GPU and OpenCL technology is presented. The developed algorithm can be easily modified to solve the MCOU problem (1), (2), where the set of uncertain factors is infinite.

The results of the computational experiment on the test task show a significant advantage of the parallel GPU implementation of the developed co-evolutionary MCOU algorithm in relation to the sequential analog.

Author Contributions: Conceptualization and methodology, V.A.S.; software and validation, D.L.P. and P.P.R.; computational experiments, E.L.D., E.Y.K. and A.V.T. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Kim, E.J.; Perez, R.E. Neuroevolutionary Control for Autonomous Soaring. *Aerospace* **2021**, *8*, 267. [CrossRef]
2. Bernas, M.; Płaczek, B.; Smyła, J. A Neuroevolutionary Approach to Controlling Traffic Signals Based on Data from Sensor Network. *Sensors* **2019**, *19*, 1776. [CrossRef] [PubMed]
3. Salichon, M.; Tumer, K. A neuro-evolutionary approach to micro aerial vehicle control. In Proceedings of the 12th Annual Genetic and Evolutionary Computation Conference (GECCO'10), Portland, OR, USA, 7–11 July 2010; pp. 1123–1130. [CrossRef]
4. Serov, V.A.; Voronov, E.M.; Kozlov, D.A. A neuroevolutionary synthesis of coordinated stable-effective compromises in hierarchical systems under conflict and uncertainty. *Procedia Comput. Sci.* **2021**, *186*, 257–268. [CrossRef]
5. Serov, V.A.; Voronov, E.M.; Kozlov, D.A. Hierarchical Neuro-Game Model of the FANET based Remote Monitoring System Resources Balancing. In *Studies in Systems, Decision and Control. Smart Electromechanical Systems. Situational Control*; Gorodetskiy A., Tarasova I., Eds.; Springer International Publishing: Berlin/Heidelberg, Germany, 2020; Volume 261, pp. 117–130. [CrossRef]
6. Serov, V.A.; Voronov, E.M.; Kozlov, D.A. Hierarchical Population Game Models of Machine Learning in Control Problems Under Conflict and Uncertainty. In *Studies in Systems, Decision and Control. Smart Electromechanical Systems. Recognition, Identification, Modeling, Measurement Systems, Sensors*; Gorodetskiy, A.E., Tarasova, I.L., Eds.; Springer: Cham, Switzerland, 2022; Volume 419, pp. 125–145. [CrossRef]
7. Serov, V.A. Hierarchical Population Game Models of Coevolution in Multi-Criteria Optimization Problems under Uncertainty. *Appl. Sci.* **2021**, *11*, 6563. [CrossRef]
8. Andión, J.M.; Arenaz, M.; Bodin, F.; Rodríguez, G.; Tourino, J. Locality-aware automatic parallelization for GPGPU with OpenHMPP directives. *Int. J. Parallel Program.* **2016**, *44*, 620–643. [CrossRef]
9. Chandrashekhar, B.N.; Sanjay, H.A. Performance Study of OpenMP and Hybrid Programming Models on CPU–GPU Cluster. In *Emerging Research in Computing, Information, Communication and Applications*; Springer: Singapore, 2019; pp. 323–337.
10. Chandrashekhar, B.N.; Sanjay, H.A. Srinivas, T. Performance Analysis of Parallel Programming Paradigms on CPU-GPU Clusters. In Proceedings of the 2021 International Conference on Artificial Intelligence and Smart Systems (ICAIS), Coimbatore, India, 25–27 March 2021; pp. 646–651.
11. Soyata, T. *GPU Parallel Program Development Using CUDA*; CRC Press: Boca Raton, FL, USA, 2018.
12. Karovič, V.; Kaźmierczakb, M.; Pankivb, O.; Górkiewiczb, M.; Zakharchukc, M.; Stolyarchukc, R. OpenCL and CUDA Comparison of MapReduce Performance on Distributed Heterogeneous Platform through Integration with Hadoop Cluster. In Proceedings of the CEUR Workshop Proceedings, IT&AS'2021: Symposium on Information Technologies & Applied Sciences, Bratislava, Slovakia, 5 March 2021; pp. 202–208.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.