Inference Method and Parallel Implementation for MISO Structure Systems for Inputs with Linguistic Values*

V. G. Sinuk, S. A. Karatach

Belgorod state technological university named after V. G. Shukhov, Belgorod, Russia

Abstract. In fuzzy modeling, both clear and fuzzy values can be given to the inputs of the simulated systems. The computational complexity of fuzzy inference with fuzzy inputs, which are a formalization of linguistic values, corresponds to exponential complexity. This paper describes a new method of inference based on the decomposition theorem of multidimensional fuzzy implication and fuzzy truth value. This method makes it possible for fuzzy inputs to implement an inference with polynomial computational complexity, which makes it effective for modeling large-dimensional MISO structure systems. The implementation of this method using parallel computing technologies is reviewed in detail. As a result of the experiment, conclusions were made about the feasibility of using a particular implementation, depending on the amount of input data.

Keywords: a logical type of inference, a decomposition theorem, a fuzzy truth value, parallel computations.

DOI 10.14357/20718632200308

Introduction

There are various fuzzy inference methods that can be divided into three types: logical inference, Mamdani-type inference, and Takagi-Sugeno-type inference [1]. All these methods are intended for modeling systems with clear input values. For systems with many fuzzy inputs that are consequences of formalization of terms of linguistic variables [3, 6], these methods are not applicable in practical problems due to exponential complexity.

This article develops a method for logical type inference if the input data are linguistic values that are formalized by membership functions.

The method is based on the decomposition theorem applicable when the composite premise of a rule is modeled by the t-norm – *min*. This theorem makes it possible to reduce the computational complexity of the inference to polynomial. Improvement of this method is carried out by applying a fuzzy truth value, the definition of which, for some accessory, functions are performed analytically [4], and in the case of a piecewise linear representation of the accessory function, an effective algorithm is developed [5].

Much attention is paid to the implementation where a comparative analysis is made of two approaches to the implementation of the method: single-thread, designed to work on the CPU, and parallel, using the CUDA technology. In addition, issues related to the implementation of a particular approach while taking into account the specifics of the target hardware platform are considered.

* The work was carried out with the financial support of the RFBR (project no. 20-07-00030).

1. Problem

The linguistic model is a knowledge base of fuzzy rules $R_k, k = \overline{1,N}$ of the form:

$$R_k: If\ x_1\ is\ A_{1k}\ and\ x_2\ is\ A_{2k}\ and, \dots, and\ x_n\ is\ A_{nk},\ then\ y\ is\ B_k \quad (1)$$

where N is the number of fuzzy rules, $A_{ik} \subseteq X_i, i = \overline{1,n}, B_k \subseteq Y$ are fuzzy sets that are characterized by membership functions $\mu_{A_{ik}}(x_i)$ and $\mu_{B_k}(y)$ respectively for $i = \overline{1,n}$. $x_1, x_2, \dots, x_n$ – are linguistic variables of the linguistic model, and $[x_1, x_2, \dots, x_n]^T = \boldsymbol{x} \in X_1 \times X_2 \times \dots \times X_n$. The symbols X_i and Y represents the spaces of input and output variables, respectively. If we denote $\boldsymbol{X} = X_1 \times X_2 \times \dots \times X_n$ and $\boldsymbol{A_k} = A_{1k} \times A_{2k} \times \dots \times A_{nk}$, then rule (1) is represented as a fuzzy implication:

$$R_k: \boldsymbol{A_k} \to B_k.$$

The R_k rule can be formalized as a fuzzy relation defined on the set $\boldsymbol{X} \times Y$, i. e. $R_k \subseteq \boldsymbol{X} \times Y$ is a fuzzy set with the membership function:

$$\boldsymbol{\mu_{R_k}}(\boldsymbol{x}, y) = \boldsymbol{\mu_{A_k \to B_k}}(\boldsymbol{x}, y) = I\left(\boldsymbol{\mu_{A_k}}(\boldsymbol{x}), \mu_{B_k}(y)\right),$$

where $I(*)$ is a fuzzy implication.

The task is to determine the fuzzy output $B'_k \subseteq Y$ for the system shown in (1), if the inputs of a fuzzy set $\boldsymbol{A'} = A'_1 \times \dots \times A'_n$ or x_1 is A'_1 and … and x_n is A'_n.

According to the generalized fuzzy rule modus ponens [3], the fuzzy set B'_k is defined by the combination of the fuzzy set $\boldsymbol{A'}$ and the relation R_k, i. e.

$$B'_k = \boldsymbol{A'} \circ (\boldsymbol{A_k} \to B_k). \quad (2)$$

The complexity of expression (2) is exponential with respect to n, i. e. $O(|X|^n \times |Y|)$.

2. Method of Inference Based on Fuzzy Truth Value and Decomposition Theorem

A special case of a composite output rule is the generalized modus ponens rule, which for systems with a single input is described by the relation [1]:

$$\mu_{B'}(y) = \sup_{x \in X}\left\{\mu_{A'}(x) \overset{T}{*} I(\mu_A(x), \mu_B(y))\right\}, \quad (3)$$

where $\mu_{A'}(x)$, $\mu_A(x)$, $\mu_{B'}(y)$, $\mu_B(y)$ are membership functions, $\overset{T}{*}$ – t-norm representing the intersection of a fuzzy fact A' and a fuzzy implication I, whose argument is the premise A and the output B. Fuzzy sets are described on the reasoning space X for the premise and fact, and on Y for the value B and the inference result B'.

Using the true modification rule [6], we can write:

$$\mu_{A'}(x) = \tau_{A/A'}\left(\mu_A(x)\right),$$

where $\tau_{A/A'}(*)$ is the fuzzy truth value of a fuzzy set A relative to A', which represents the membership function of the compatibility $CP(A, A')$ A with respect to A', and A' is considered reliable [7].

$$\tau_{A/A'}(t) = \mu_{CP(A,A')}(t) = \sup_{\substack{\mu_A(x)=t \\ x \in X}} [\mu_{A'}(x)], \quad t \in [0,1] \quad (4)$$

When passing from variable x to variable t, denoting $t = \mu_A(x)$ we get:

$$\mu_{A'}(x) = \tau_{A/A'}\left(\mu_A(x)\right) = \tau_{A/A'}(t)$$

Then (3) is written as follows:

$$\mu_{B'}(y) = \sup_{t \in [0,1]}\left\{\tau_{A/A'}(t) \overset{T}{*} I(t, \mu_B(y))\right\}. \quad (5)$$

If the membership function $\mu_A(x)$ and $\mu_{A'}(x)$ are given as Gaussian curves or as a bell-shaped function, then, as follows from [4], the fuzzy truth value (4) is determined analytically.

In the case of piecewise linear representation of membership functions, an algorithm with polynomial computational complexity is developed [5].

Consider the decomposition theorem of a multidimensional fuzzy implication. The theorem can be used if the modeling of the linguistic conjunction «*and*» in the antecedent of rule (1) applies the t-norm – *min*.

Theorem

If $I\left[\mu_{A_{ik}}(x_i), \mu_{B_k}(y)\right], i = \overline{1,n}$ does not increase by the argument $\mu_{A_{ik}}(x_i)$, then:

$$I\left(\boldsymbol{\mu_{A_k}}(\boldsymbol{x}), \mu_{B_k}(y)\right) = I\left(\min_{i=\overline{1,n}}\{\mu_{A_{ik}}(x_i)\}, \mu_{B_k}(y)\right) = \max_{i=\overline{1,n}}\{I(\mu_{A_{ik}}(x_i), \mu_{B_k}(y))\}.$$

Let's prove it:

Definition

The function $I(\mu_{A_{ik}}(x_i), \mu_{B_k}(y))$ is non-growing by the argument $\mu_{A_{ik}}(x_i)$, if $\forall \mu^*_{B_k}(y) \in [0,1]$ from the condition $\mu^l_{A_{ik}}(x_i) > \mu^m_{A_{ik}}(x_i)$, follows the inequality:

$$I(\mu^l_{A_{ik}}(x_i), \mu^*_{B_k}(y)) \le I(\mu^m_{A_{ik}}(x_i), \mu^*_{B_k}(y)) \quad (6)$$

Let's choose an arbitrary set of argument values $(x_1^*, x_2^*, \dots, x_n^*) \in X_1 \times X_2 \times \dots \times X_n$ and define the antecedent of rules (1) for these values, when the linguistic conjunction «*and*» is modeled by the «*min*» operation:

$$\mu_{A_{jk}}(x_j^*) = \min_{i=\overline{1,n}}\{\mu_{A_{ik}}(x_i^*)\}, \quad (7)$$

It follows that: $\mu_{A_{jk}}(x_j^*) \le \mu_{A_{ik}}(x_i^*), \forall i = \overline{1,n}$. According to definition (6), the function $I(*)$ is non-growing. Thus, we have:

$$I\left(\mu_{A_{jk}}(x_j^*), \mu_{B_k}^*(y)\right) \ge I\left(\mu_{A_{ik}}(x_i^*), \mu_{B_k}^*(y)\right), \quad \forall i = \overline{1,n}, \quad (8)$$

Then, given the ratio (8):

$$I\left(\mu_{A_{jk}}(x_j^*), \mu_{B_k}^*(y)\right) = \max_{i=\overline{1,n}}\left\{I\left(\mu_{A_{ik}}(x_i^*), \mu_{B_k}^*(y)\right)\right\} \quad (9)$$

Since $I\left(\mu_{A_{jk}}(x_j^*), \mu_{B_k}^*(y)\right)$ exists on the right side of expression (8), taking into account (7), we will rewrite (9) as:

$$I\left(\min_{i=\overline{1,n}}\{\mu_{A_{ik}}(x_i^*)\}, \mu_{B_k}^*(y)\right) = \max_{i=\overline{1,n}}\{I(\mu_{A_{ik}}(x_i^*), \mu_{B_k}^*(y))\} \quad (10)$$

The resulting expression (10) is valid for $\forall \mu_{B_k}^*(y) \in [0,1]$ and $\forall (x_1^*, x_2^*, \dots, x_n^*) \in X_1 \times X_2 \times \dots \times X_n$.

The theorem is proved.

For a system with multiple inputs (3) has the form:

$$\mu_{B'_k} = \sup_{\boldsymbol{x} \in \boldsymbol{X}}\left\{\boldsymbol{\mu}_{\boldsymbol{A'}}(\boldsymbol{x}) * I\left(\boldsymbol{\mu}_{\boldsymbol{A_k}}(\boldsymbol{x}), \mu_{B_k}(y)\right)\right\} \quad (11)$$

If the condition of the decomposition theorem of a multidimensional fuzzy implication of non-growing $I(\mu_{A_{ik}}(x_i), \mu_B(y))$, $i = \overline{1,n}$ relative to $\mu_{A_{ik}}(x_i)$ and the use of the linguistic conjunction «*and*» (11) takes the form:

$$\mu_{B'_k}(y) = \max_{i=1,n}\left\{\sup_{x_i \in X_i}\left\{\mu_{A'_i}(x_i)^T * I(\mu_{A_{ik}}(x_i), \mu_{B_k}(y))\right\}\right\} \quad (12)$$

Expression (12) can be written using a fuzzy truth value of the degree of truth, as follows from (4), i. e. (12) will have the form:

$$\mu_{B'_k}(y) = \max_{i=1,n}\left\{\sup_{t_i \in [0,1]}\left\{\tau_{A_{ik}/A'_i}(t_i)^T * I(t_i, \mu_{B_k}(y))\right\}\right\} \quad (13)$$

The discrete analog of the relation (13) has polynomial computational complexity, i. e. $O(|t_i| \times |Y| \times n)$. The non-growing condition $I\left(t_i, \mu_{B_k}(y)\right), i = \overline{1,n}$ with respect to t_i is satisfied for some implications [2].

3. Inference for the Rule Base

Consider the output values for N rules for the form (1) using the center of gravity defuzzification method [1]:

$$\bar{y} = \frac{\sum_{k=\overline{1,N}} \bar{y}_k * \mu_{B'}(\bar{y}_k)}{\sum_{k=\overline{1,N}} \mu_{B'}(\bar{y}_k)} \quad (14)$$

where $\bar{y}$ is a clear output of a system consisting of N rules of the form (1); $\bar{y}_k, k = \overline{1,N}$ are the centers of the membership functions $\mu_{B_k}(y)$, i. e. the values in which $\max_y \mu_{B_k}(y) = 1$. B' is obtained by a logical approach using the intersection operation in accordance with the expression:

$$B' = \bigcap_{j=\overline{1,N}} B'_j$$

The membership function B' is calculated using t-norm, that is:

$$\mu_{B'}(y) = \mathop{\mathrm{T}}_{j=\overline{1,N}} \mu_{B'_j}(y) \quad (15)$$

Is the condition of non-increasing implication $I\left(t_i, \mu_{B_k}(y)\right), i = \overline{1,n}$ relative to t_i is met and the linguistic conjunction «*and*» is used in (1), then from the relations (13), (14) and (15) we get the expression:

$$\bar{y} = \frac{\sum_{k=\overline{1,N}} \bar{y}_k \mathop{\mathrm{T}}_{j=\overline{1,N}}\left\{\max_{i=\overline{1,n}}\left\{\sup_{t_i \in [0,1]}\left\{\tau_{A_{ik}/A'_i}(t_i)^T * I(t_i, \mu_{Bj}(\bar{y}_k))\right\}\right\}\right\}}{\sum_{k=\overline{1,N}} \mathop{\mathrm{T}}_{j=\overline{1,N}}\left\{\max_{i=\overline{1,n}}\left\{\sup_{t_i \in [0,1]}\left\{\tau_{A_{ik}/A'_i}(t_i)^T * I(t_i, \mu_{Bj}(\bar{y}_k))\right\}\right\}\right\}} \quad (16)$$

The relation (16) corresponds to the network structure (Fig. 1), where the expression is defined at the second level:

$$F_{ijk}\{\bullet\} = \sup_{t_i \in [0,1]}\left\{\tau_{A_{ik}/A'_i}(t_i)^T * I(t_i, \mu_{Bj}(\bar{y}_k))\right\}$$

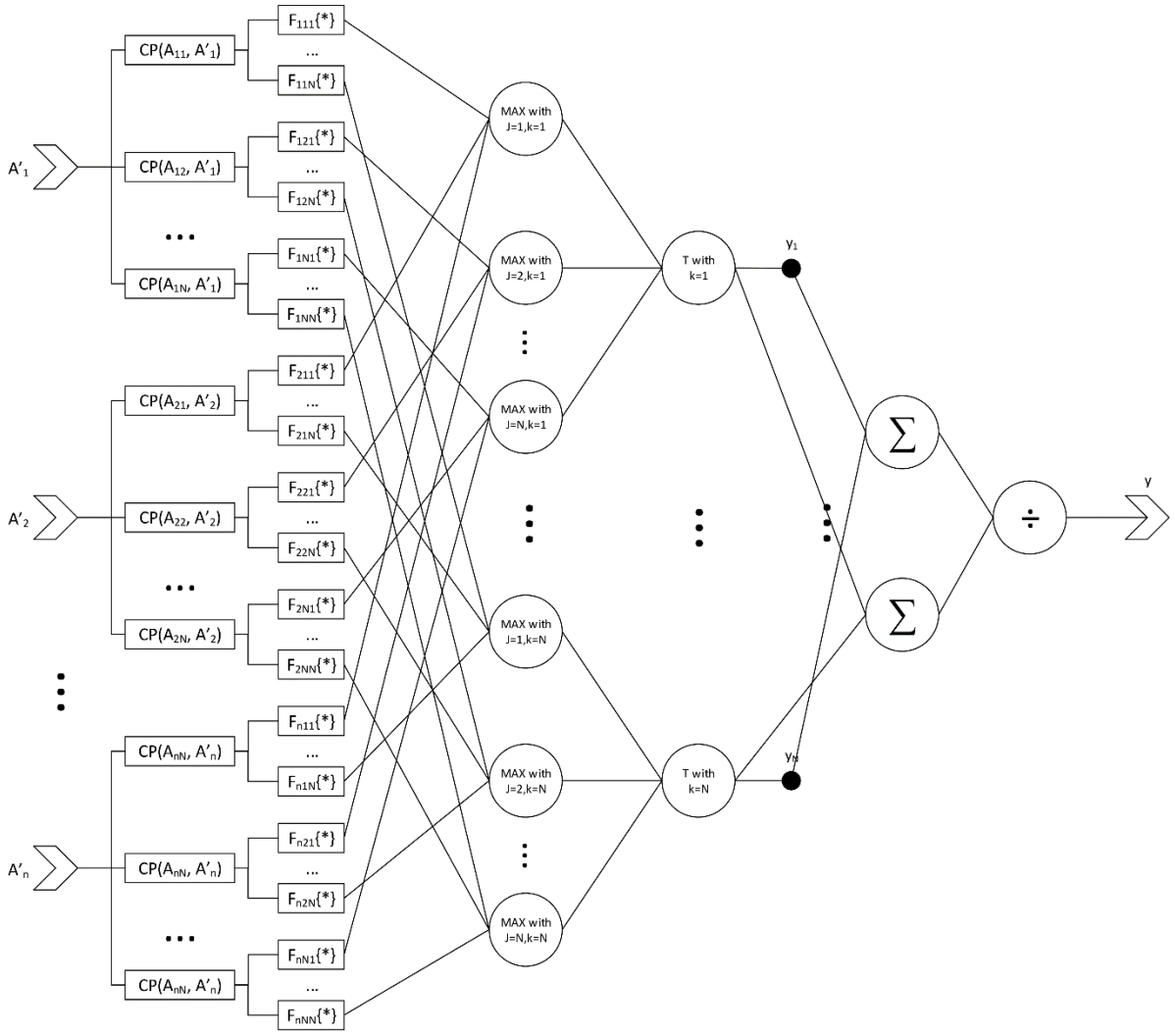

Fig. 1. Network structure for the rule base

4. Representation of a Fuzzy System in Computer Memory and Organization of Computations

The inputs of this type of fuzzy inference systems represent the values of the corresponding linguistic variables. Each linguistic variable has a valid set of its fuzzy values, called a term set. The elements of such a set are fuzzy variables defined in the same space as the linguistic variable. The value of such a fuzzy variable is determined by a fuzzy set, which is represented in the computer memory by discretizing the corresponding membership function. Whereas the entire rule base can be represented using a pair of tables of discretized term sets of linguistic variables and a table of indexes of the values of these discretized term sets. The row elements of the last table thus form a fuzzy fact. A fuzzy premise can be formed in a similar way by discretizing each of the accessory functions.

Since an additional level of indirection was introduced to represent the rule base in the form of an index table, each individual element of the term set of the table of linguistic variables can be mapped to a different space by some rule. Thus, for each of the possible values of a given linguistic variable, the center of the membership function and a fuzzy truth value for the corresponding input value can be calculated. As a result of using this order of calculations for some of them, the dependence of the time complexity function on the

size of the rule block is replaced by the dependence on the power of the term set of the linguistic variable corresponding to this input.

The efficiency of single-thread implementation is achieved both by composing effective program code that provides an optimal sequence of commands that takes into account the target configuration of the CPU, and by using the method of organizing data in memory and accessing it, based on the generally applicable concept of the memory hierarchy. The optimal set of instructions within this context of program code execution is provided by most modern compilers at the proper level. At the same time, the data structure in memory is designed based on the nature of their use, with certain restrictions on their alignment and the size of each data block.

At the same time, parallel implementation using CUDA technology involves building an application consisting of part of the code intended for execution on the CPU (host) and part for execution on the GPU (device). This requires a more specific approach for organizing efficient calculations both on the host and on the device [8-10]. In General, the host performs the necessary preparation actions before running the code on the device. For example, Fig. 2 shows a flowchart for calculations based on this algorithm, where the blocks including calculations on the GPU are highlighted in gray.

The order of the calculation steps here is the same as in the single-threaded implementation, but some of these steps include calculations on the device. So, the calculation of fuzzy truth values mentioned earlier can be performed on the GPU.

In addition, the main cycle of the algorithm consists of two parts: finding the values of $F_{ji}(y_k)$ for each input independently and convolution of the obtained values to $\mu_{B'}(\bar{y}_k)$. The CUDA technology is designed to guarantee that a number of types of operations are independent: calculations on the host, calculations on the device, copying data from the host to the device, and so on. The independence of these types of operations implies the potential for their overlapping. For example, the mechanisms provided by the CUDA software interface are used to organize simultaneous calculations on different inputs, such as streams, asynchronous data copying, and the mechanism for blocking memory pages.

To achieve asynchronous copying to be performed correctly, memory pages containing the source data must be locked in RAM. And data copying itself must be performed using streams, which is a special concept of CUDA technology.

A set of locked pages can be obtained using the cudaHostAlloc() function for the entire volume of input data. In this case, the selection is made using the cudaHostAllocWriteCombined flag, which allows you to avoid using the memory management unit of the central processor and accomplish direct data transfer to the device.

The ability to simultaneously copy portions of data and execute CUDA kernels is achieved by using streams. In this case, the portion is a set of data required for calculations for this input. Copying each such portion is performed by calling the cudaMemcpyAsync function. After this, a call to

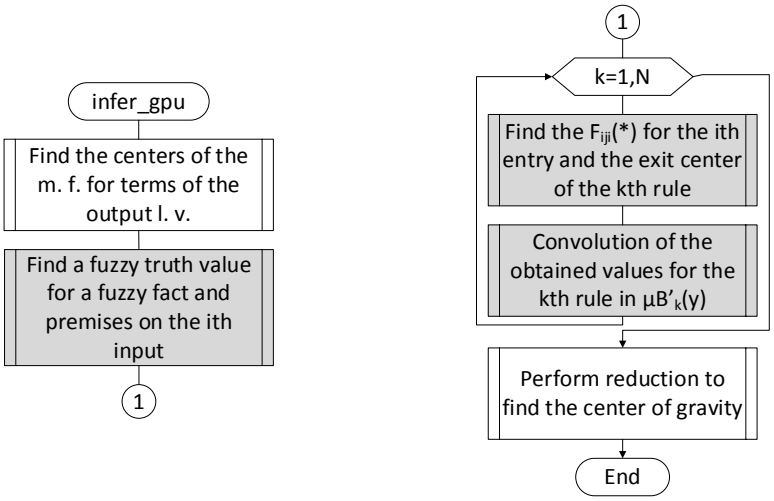

Fig. 2. A flowchart corresponding to a parallel implementation of the fuzzy output algorithm on the host side

the CUDA kernel is added to the same stream, which will be executed simultaneously with copying data for the next inputs. This method of optimizing the use of available hardware resources for a system with seven inputs is illustrated in Fig. 3.

The optimal configuration of each running CUDA core will ensure full employment of streaming multiprocessors on the device. To achieve this, the maximum possible number of thread sets must be executed on each streaming multiprocessor, which will compensate for the delays that occur when waiting for data loading or the result of the execution of the previous instruction, by instantly switching to work with a different set of threads. However, it is recommended to keep the block size as small as possible, and each streaming multiprocessor has a limit on the maximum number of blocks that can be on it at the same time, depending on the version of the architecture used for this GPU. For example, we use a device with compute capability 3.5 which has a limit on the maximum number of blocks in the multiprocessor equal to 16, and the maximum number of resident blocks equal to 64. Initially, each block contains a single set of threads, but to achieve maximum utilization of the streaming multiprocessor, the concept of “executor” was introduced, according to which the block size is chosen as the minimum necessary for full occupancy and is divided into groups of warps called “executors”. The size of each such group is equal to the original block size. This approach allows you to load all streaming multiprocessors with as many threads as possible.

To reduce the duration of data access delays that occur during the operation of CUDA kernels, it is common practice to prefer place data in shared memory as opposed to global memory. This means that data is loaded into shared memory from the global memory, processed, and uploaded back to the global memory. However, the use of shared memory is justified if the data is modified or there is a need to store intermediate results. When data in global memory is accessed solely for the purpose of retrieving their values, there is a mechanism for automatically caching this data. Data that is immutable throughout the CUDA kernel can be manually stored in the read-only data cache by reading it using the __ldg () function. The same effect can be achieved by marking the pointer to such data with the const and __restrict__modifiers. So, to find the maximum implication value, each "executor" is allocated a buffer of the required size, in which parallel convolution is then performed, and pointers to the tables of fuzzy sets, fuzzy truth values, and the index table are marked with the const and __restrict__modifiers.

In addition to memory access delays, the performance of CUDA kernels is negatively affected by the presence of divergent execution branches that occur when the execution flow is parameterized by conditional constructs. In some situations,

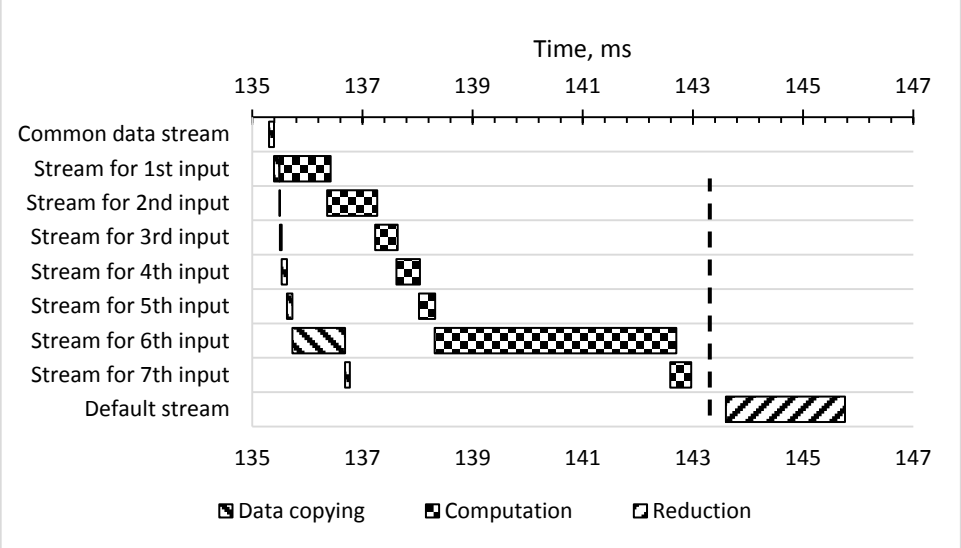

Fig. 3. Overlapped copying of data and calculations to the GPU using streams

Table 1. Influence of the method of organizing high-level calculations on the possibility of diverging execution branches

for (i = 0; i < length(a) - 1; ++i) {
  float a1 = a[i];
  float a2 = a[i+1];
 	
  if (a1 <= t && t <= a2) {
  float y = /* … */;
  if (y > ftv) ftv = y;
  }
}
	for (i = 0; i < length(a) - 1; ++i) {
	  float a1 = a[i];
	  float a2 = a[i+1];
	
	  float y = /* … */;
	  if (a1 <= t && t <= a2 && y > ftv) {
	  ftv = y;
	  }
	}

There is a divergence.	There is no divergence (the conditional assignment statement is used).

when actions in the body of such blocks can be reduced to a single data-modifying instruction, the compiler can replace the conditional jump through the instruction block that generates the divergence with a predicate instruction that performs data modification if the condition is true. For example, when calculating a fuzzy truth value for a given node of the coordinate grid, you should give preference to performing extra calculations, while ensuring that the code in the body of the loop is linear (as in the code fragment to the right of the two shown in Table 1).

Thus, to achieve maximum performance in a parallel approach to performing calculations, the following principles of optimization of CUDA applications were used:

– simultaneous data copying and code execution on the GPU.

– maximization of employment (occupation) of computing blocks.

– data caching.

– getting rid of diverging branches of execution.

5. The Results of Computational Experiment

A test data set taken from the UC Irvine Machine Learning Repository electronic resource was selected to perform computational experiments [11]. The set contains 45211 entries. The data is related with direct marketing campaigns of a Portuguese banking institution. The marketing campaigns were based on phone calls. Often, more than one contact to the same client was required, in order to access if the product (bank term deposit) would be ('yes') or not ('no') subscribed. The purpose of classification is to predict whether the client will subscribe ("yes"/"no") a term deposit.

Thus, the system has the following list of input variables (attributes of the bank's client): age (numeric), type of activity (categorical), marital status (categorical), availability of another loan (categorical), average annual balance in euros (numeric), availability of a mortgage (categorical), has a private loan (categorical), selected method of communication (categorical), and so on. The system also has a single output variable: whether the client will subscribe a term deposit (binary).

A program in Python was written for performing computational experiments. This program consists of two stages: preparing the source data and directly measuring the execution time of a specific implementation of the output algorithm. The implementations themselves were written in C/C++ with the CUDA extension and compiled into machine code to achieve the maximum level of control over hardware resources.

The process of preparing source data consists of building a database of rules. For this purpose, each of the parameters described above is considered as a linguistic variable, the list of values of which is made up of a set of categories associated with each parameter. In the case of numeric parameters, the range of their values is divided into sub-ranges, each of which is treated as a category. The dimension of the base set of each linguistic variable coincides with the dimension of the set of values. And each of its values is assigned a membership function, with a peak value at the point corresponding to it.

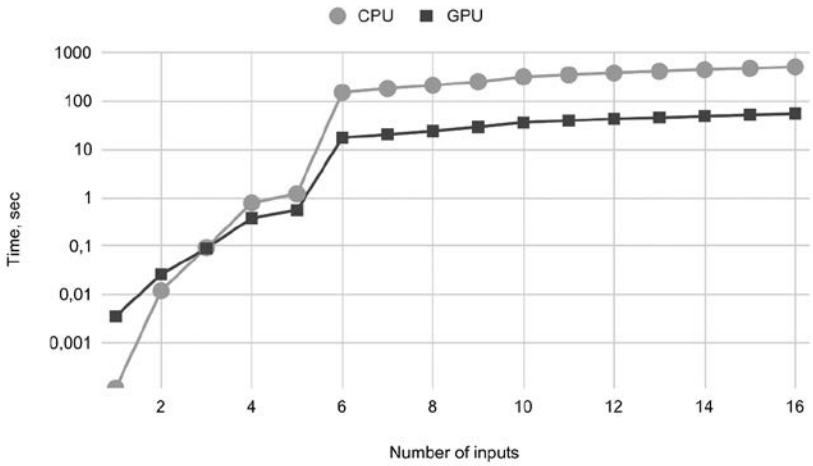

Fig. 4. Execution time graph for two implementations depending on the number of inputs

Next, computational experiments were performed for n=1.16 inputs. The calculations were performed on a device with a Central processor – Intel (R) Core(TM) i7-3930K CPU @ 3.20 GHz and a graphics processor – Nvidia Titan K20c. The result of the experiments is shown in the Fig. 4.

After analyzing the graphs in the figure, you can give a number of comments explaining their form.

When the number of inputs is up to four, there is an advantage in execution time for the single-threaded implementation of the algorithm. This situation in relation to a parallel implementation is related to the need to copy data to a range of locked pages, the locking of which is a system call, and may take a different amount of time, depending on the implementation of a particular operating system. In addition, there is latency (the running time of the algorithm for zero data volume) when preparing for the launch of the CUDA kernel and performing launch itself.

With more inputs, there is a sharp "leap" in execution time for both single-threaded and parallel implementations. In the case of a single-threaded implementation, this can be explained by the relatively small amount of L1 cache available to a single processor core. In this case, if the base data set is large enough, i.e. data that is referenced by other data structures, there will be a large number of cache misses. A similar situation with bandwidth restriction can occur on the GPU. In addition, a high level of difference in the dimensions of fuzzy sets for each specific input can also contribute a fraction of the distortion to the final graph.

Thus, the choice between single-thread or parallel implementation depends on the characteristics of the hardware used and must take into account the latency when accessing the GPU driver.

Conclusion

The inference based on the decomposition theorem makes it possible to extend the logical approach for systems with many fuzzy inputs, which is a formalization of the terms of a linguistic variable, with polynomial computational complexity.

This paper also describes a single-threaded and parallel method for implementing the fuzzy inference method based on the fuzzy truth value and the decomposition theorem. For each method, a list of key points for organizing calculations and working with memory that allow you to achieve maximum performance of the final implementation is provided. Then computational experiments were performed to compare the running time of a particular implementation of the algorithm for different amounts of input data. As a result of the experiment, conclusions were made about the feasibility of using a particular implementation, depending on the volume of input data.

References

1. Rutkowski L. 2009. Methods and techniques of artificial intelligence. PWN. 452 p.
2. V. G. Sinuk, M. V. Panchenko. 2017. Methody nechetkogo vivoda dlya odnogo klassa system MISO-struktury pri nechetkih vhodah [Fuzzy inference method for a one class of MISO structure systems with fuzzy in-

puts.]. Iskustvenniy intellect i prinyatie resheniy [Artificial intelligence and decision making]. 4:33-39.
3. Zadeh L. 1976. Ponyatie lingvisticheskoy peremennoy i ego primenenie k prinyatiyu pribligennih resheniy [The concept of a linguistic variable and its application to approximate decision making]. Mir. 168 p.
4. V. G. Sinuk, E. V. Pivnenko. 2006. Ob analiticheskom vichislenii nechetkogo znacheniya istinnosti [On analytical calculation of a fuzzy truth value]. Sbornik trudov Vserossiyskoy nauchnoy konferencii po nechetkim sistemam i myagkim vichisleniyam (NSMV-2006)[Proceedings of the all-Russian scientific conference on fuzzy systems and soft computing (NSMV-2006)]. 129-133.
5. D. A. Kucenko, V. G. Sinuk. 2008. Algoritmi nahogdeniya CP pri kusochno-lineynom predstavlenii funkciy prinadlegnosti[Algorithms for finding CP in piecewise linear representation of membership functions]. Sbornik trudov vtoroy Vserossiyskoy nauchnoy konferencii po nechetkim sistemam I myagkim vichisleniyam (NSMV-2008) [Proceedings of the second all-Russian scientific conference on fuzzy systems and soft computing (NSMV-2008)]. 87-92.
6. A. N. Borisov, A. V. Alekseev, O. A. Krumberg. 1982. Decision models based on a linguistic variable. Riga: Zinatne. 256 p.
7. D. Dobua, A. Prad. 1990. Theoriya vozmognostey. Prilogenie k predstavleniyu znaniy v informatike[Theory of possibilities. Applications to knowledge representation in computer science]. Moskow: Radio and communications. 228 p.
8. Jason Sanders, Edward Kandrot. 2010. CUDA by example: An introduction to general-purpose GPU programming. Addison-Wesley Professional. 320 p.
9. CUDA Programming Guide. Available at: https://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html (accessed July 3, 2020)
10. CUDA Best Practices Guide. Available at: https://docs.nvidia.com/cuda/cuda-c-best-practices-guide/index.html (accessed July 3, 2020)
11. Bank Marketing Data Set. Available at: https://archive.ics.uci.edu/ml/datasets/Bank+Marketing (accessed July 3, 2020)

Sinuk V. G. Professor, Belgorod state technological university named after V. G. Shukhov, 46 Kostukova str., Belgorod, 308012, Russia, email: vgsinuk@mail.ru

Karatach S. A. Student, Belgorod state technological university named after V. G. Shukhov, 46 Kostukova str., Belgorod, 308012, Russia, email: karatach1998@yandex.ru