УДК 681.3.06

С.В. Єршов, Р.М. Пономаренко

МЕТОД ПОБУДОВИ ПАРАЛЕЛЬНИХ СИСТЕМ НЕЧІТКОГО ЛОГІЧНОГО ВИВЕДЕННЯ НА ОСНОВІ ГРАФІЧНИХ ПРИСКОРЮВАЧІВ

Розроблено та обґрунтовано метод побудови паралельних систем нечіткого логічного виведення Такагі – Сугено на основі графічних прискорювачів Nvidia. Розроблено інтелектуальну систему оцінювання якості стартапів на основі ієрархічних систем нечіткого логічного виведення Такагі – Сугено. Встановлено оцінки прискорення систем нечіткого виведення, що побудовані за зазначеним методом, здійснено порівняльну характеристику з варіантами нечітких систем, що реалізовані на базі технології паралельного програмування MPI.

Ключові слова: інтелектуальна система, ярусно-паралельна форма обчислень, графічні прискорювачі, система Такагі – Сугено, нечітка логіка, CUDA.

Вступ

У даний час нечітка логіка широко застосовується у багатьох сферах при створенні інтелектуальних програмних систем та технологій, а саме, діагностичних систем, систем розпізнавання образів, інтелектуальних мультиагентних систем, що забезпечують підтримку прийняття складних рішень в умовах невизначеності [1, 2]. Методи нечіткого виведення, що використовуються у різних інтелектуальних системах, грунтуються на базах знань, які задаються у вигляді сукупності нечітких предикатних правил. Запропоновано кілька методів нечіткого логічного виведення, які відрізняються операторами застосування та композиції нечітких множин, що містяться у правилах, методами приведення отриманих нечітких значень до чітких чисел, таких як методи Мамдані, Ларсена, Такагі – Сугено та інші [2]. Зокрема, нечіткий метод Такагі – Сугено ґрунтується на базі знань, що представляється наборами правил «якщо-то», які можуть бути узагальнені наступним чином:

R_j : якщо $x_1 \in A_{1j}$ та $x_2 \in A_{2j}$ та …

… та $x_n \in A_{nj}$ то $y = g_j(x_1, x_2, ..., x_n)$,

$j = 1,2,..., N$,

де g_j – чітка функція від x_i.

Проте, при побудові інтелектуальних систем існує проблема, що заважає будувати нечіткі системи великої розмірності, а саме, при збільшенні кількості вхідних значень у системі значно збільшується кількість правил. Система з нечіткими правилами (СНП) з n вхідними змінними та l лінгвістичними термами на одну змінну містить l^n правил [3]. Очевидно, що побудувати та перевірити таку кількість правил для систем, що мають кілька десятків вхідних змінних, не під силу навіть досвідченим експертам. Тому для вирішення задач великої складності та розмірності запропоновані ієрархічні системи нечіткого логічного виведення, що покликані зменшити число правил системи та відповідно збільшити складність систем логічного виведення на основі нечіткої математики.

Запропоновано різні варіанти введення ієрархічних структур для систем з нечіткими правилами [3–7]. Правила або змінні можуть бути розподілені на різних рівнях відповідно до їх специфічності, деталізації, релевантності тощо. Визначення ієрархічних нечітких систем як методу вирішення задач нечіткого логічного виведення з вищим рівнем складності, ніж ті, для яких зазвичай призначені звичайні СНП, базується на застосуванні певного способу декомпозиції, що породжує ієрархію систем нижчої складності [3].

Перший метод побудови ієрархічних структур визначає такі структури на основі пріоритетності правил таким чином,

що правила з неоднаковим рівнем специфічності отримують неоднаковий пріоритет. При цьому, правила, що мають більш високий пріоритет, є більш специфічними [4, 5]. Загальне правило застосовується лише в тих випадках, коли не існує відповідного специфічного правила. У цьому випадку ієрархія утворюється в результаті застосування конкретного механізму імплікації, який застосовує правила, беручи до уваги їх пріоритет. Для розробки ієрархічних СНП правила мають бути згруповані за певними рівнями пріоритетів.

Інший варіант полягає у тому, щоб розглянути ієрархію розбиття вхідного простору змінних з різною деталізацією [6]. Нечіткі розбиття описують набори лінгвістичних термів, пов'язаних з кожною змінною в лінгвістичних правилах, а також функції належності, що визначають семантику зазначених лінгвістичних термів. З цієї точки зору, СНП може бути структуровано за шарами, де кожен шар містить нечіткі розбиття з окремою деталізацією, а також правила, що використовують ці нечіткі розбиття. Зазвичай, кожне розбиття у певному шарі має однакове число нечітких термів. У цьому випадку правила, що належать до різних шарів мають різний рівень деталізації, що певним чином пов'язано з вищезазначеною ідеєю специфічності.

Зовсім інший метод – це введення декомпозиції на рівні змінних. У цьому випадку вхідний простір розбивається на підпростори нижчої розмірності, і кожна вхідна змінна розглядається тільки на певному рівні ієрархії. Результатом є каскадна структура СНП, в якій, крім підмножини вхідних змінних, вихід кожного рівня розглядається як один з можливих входів наступного рівня [7]. У результаті нечітка система розкладається на скінчене число підсистем зменшеної складності, усуваючи необхідність застосування надскладної машини логічного виведення. Така декомпозиція зазвичай виражається як спосіб усунути проблеми, що виникають внаслідок так званого зростання розмірності, тобто експоненціального зростання кількості правил, пов'язаних з кількістю змінних нечіткої системи.

У такому підході до ієрархічних СНП змінні (і правила) поділяються на різні рівні таким чином, що найбільш впливові змінні обрані як вхідні змінні на першому рівні, наступні найбільш важливі змінні вибираються як вхідні змінні на другому рівні тощо. Вихідні змінні окремих рівнів вводяться як вхідні змінні на наступному рівні.

З такою структурою правила на першому рівні СНП мають структуру аналогічну будь-якій СНП типу Такагі – Сугено або Мамдані, але на k-му рівні ($k > 1$), правила містять як вхід вихідні значення попереднього рівня

$$IF\ X_{Nk+1}=LT_{Nk+1}\ and\ \ldots\ and\ X_{Nk+nk}=LT_{Nk+nk}$$

$$and\ Ok\text{-}1=LTO_{k-1}\ THEN\ O_k=LT_{Ok},$$

де значення N_k визначає кількість вхідних змінних, що розглядаються на попередніх рівнях

$$N_k = \sum_{t=1}^{k-1} n_t ,$$

де n_t – кількість змінних системи, застосованих на рівні t. Змінна Ok представляє вихідне значення k-го рівня ієрархії. Всі вихідні значення – це проміжні змінні, за винятком виходу останнього рівня Y (загальне вихідне значення нечіткої системи).

За допомогою зазначеної структури показано [7], що кількість правил у повній базі правил може бути представлена як лінійна функція від числа змінних, тоді як у звичайних (неієрархічних) СНП достатня кількість правил визначається експоненціальною функцією від числа змінних.

Однак, застосування вищезазначеного підходу представлено в [2] тільки для систем нечіткого виведення, для яких кількість рівнів не перевищує 2–3, а кількість вхідних змінних – не більше кількох десятків відповідно. Подальше збільшення складності ієрархічних систем нечіткого виведення вимагає розв'язання проблеми, що пов'язана із значним збільшенням часу виконання логічного виведення

в складних моделях з великою кількістю ієрархічних нечітких систем.

Для подолання зазначеної проблеми запропоновано методи паралельної роботи нечітких ієрархічних систем для зменшення часу їх виконання з використанням технології MPI [8, 9].

Метою даної роботи є розробка методу побудови паралельних алгоритмів нечіткого логічного виведення на базі сучасної архітектури графічних прискорювачів та технології CUDA для отримання суттєво більшого прискорення, ніж на основі MPI, та встановлення показників ефективності складних ієрархічних систем нечіткого логічного виведення Такагі – Сугено, побудованих на основі ярусно-паралельних моделей обчислень.

Ярусно-паралельна модель обчислень

У роботах [8, 9] розроблено та обґрунтовано алгоритми ярусно-паралельної схеми обчислень для ієрархічних систем нечіткого логічного виведення.

В даній роботі для здійснення нечіткого логічного виведення на графічних прискорювачах, за основу було взято саме ярусно-паралельну форму обчислень, проте, як буде показано далі, в даній модифікації буде реалізовано внутрішній паралелізм кожної окремої вершини графа залежностей ієрархічної системи.

Побудова графа залежностей є основою аналізу паралельних алгоритмів. Будь-який паралельний алгоритм можна представити у вигляді орієнтованого ациклічного мультиграфа $G = (V, E)$, де V – множина вершин, а E – множина ребер графа. Вершини представляють множину операцій алгоритму, кожна операція алгоритму має власні вхідні аргументи, та не може бути виконана до моменту виконання всіх операцій-постачальників аргументів для неї.

Нехай u та v – пара вершин графа. Наприклад, припустимо, що вершина v має постачати обчислені аргументи вершині u. Тоді існує ребро графа від вершини v до вершини u. Якщо ж вершина v не постачає аргументів (змінних) для вершини u, то дугу не проводимо. Вершини, що не мають вхідних (вихідних) дуг називаються вхідними (вихідними) вершинами відповідно, а весь граф – графом алгоритму. У випадку, коли дуги (залежності між операціями) відсутні, аргументи операцій постачаються із вхідних даних. Під операцією розуміємо множину логічно взаємозв'язаних інструкцій програми [10].

Введемо поняття ярусу паралельної форми алгоритму. Нехай задано орієнтований ациклічний граф, який має n вершин. Кожну вершину даного графа можна помітити таким числом, меншим за n, що якщо з вершини з індексом i іде дуга з індексом j, то $i < j$ [10]. Група вершин, які мають однаковий індекс, називається ярусом паралельної алгоритму, а число вершин i-ї групи – шириною ярусу. Наступний ярус не розпочне роботу до тих пір, поки не закінчиться виконання останньої операції попереднього ярусу, тобто не буде проведена ярусна синхронізація.

Паралельні алгоритми, які досліджуються в даній роботі, відповідають ярусно-паралельній формі алгоритмів [10] (рис. 1). У даному випадку N=7, $s = 3$. Алгоритм ярусно-паралельної форми виконується наступним чином:

1) обчислюються результати вершин першого ярусу (i =1) до тих пір, поки не відпрацюють всі до останньої вершини даної групи, тобто не буде проведена синхронізація роботи вершин ярусу 1;

2) попередній ярус відкидається, повторюються операції з першого пункту, але вже для ярусу $i = i + 1$;

3) повторюється (2) до тих пір, поки $i <= s$. При досягненні значення $i > s$ процес обчислень за ярусно-паралельною формою вважається завершеним.

Процес синхронізації за ярусами є обов'язковою умовою ярусно-паралельної форми обчислень. Проте, якщо блоки правил у вершинах графа залежностей мають приблизно однаковий час роботи, на прискорення всієї ієрархічної системи синхронізація суттєво не впливає.

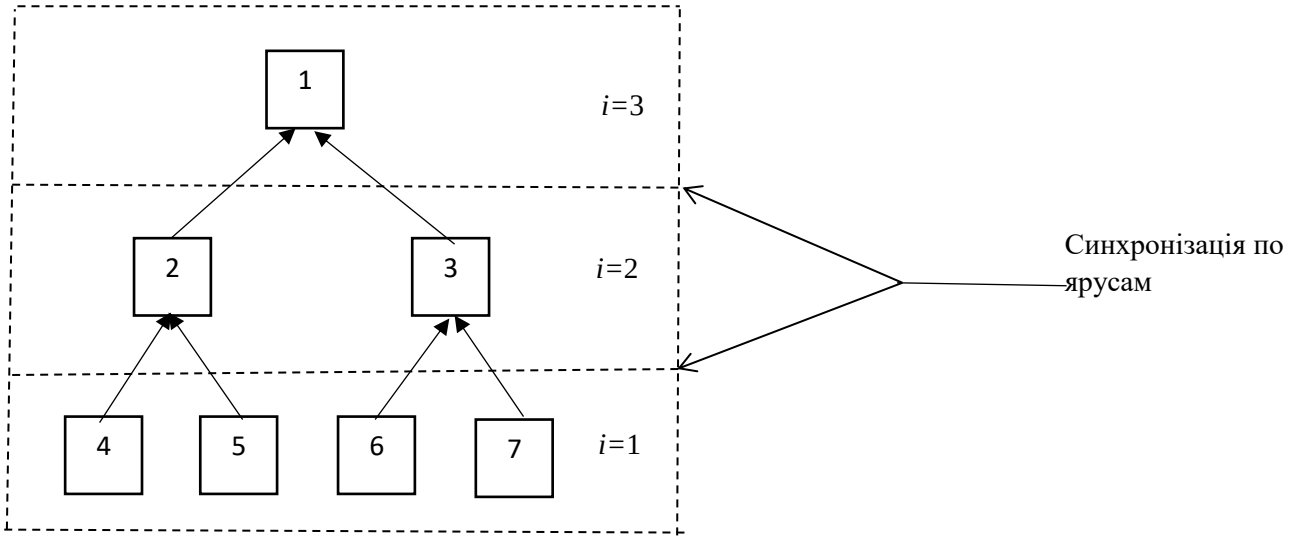

Рис. 1. Схема ярусно-паралельного алгоритму

Паралельний метод нечіткого логічного виведення Такагі – Сугено на основі архітектури CUDA

Технологія CUDA [11] – це технологія, що дозволяє використовувати відеокарту як обчислювальний пристрій з метою прискорення виконання обчислень. Керування всією програмою залишається за процесором, на GPU виконуються тільки окремі функції, команду до запуску яких надає центральний процесор. Завантаження та вивантаження даних із пам'яті GPU до оперативної пам'яті також покладено на неї.

Серед труднощів роботи з технологією CUDA можна виділити обмеження розмірності вкладених циклів. Методи усунення даної проблеми запропоновано в [12].

Паралельний метод нечіткого логічного виведення Такагі – Сугено грунтується на ярусно-паралельній схемі обчислень.

Побудова ярусно-паралельної схеми обчислень. Першим етапом є побудова ярусно-паралельної схеми обчислень для ієрархічної нечіткої системи. У випадку ієрархічних нечітких систем множиною вершин вважатимемо множину нечітких блоків правил. Зв'язки між вершинами (залежності між нечіткими блоками правил, елементарними системами нечіткого виведення), або ребра графу залежностей задаються на етапі введення вхідних даних.

Ярусно-паралельна схема містить наступні етапи обчислень (i – номер ярусу, s – кількість ярусів, n – кількість вершин у графі залежностей):

1) знаходимо всі вершини, що не мають дуг (висячі), та записуємо їх індекси до списку групи вершин першого ярусу (i=1). Змінну s інкрементуємо на одиницю;

2) відкидаємо з графу вершини, знайдені на попередньому етапі та шукаємо за попереднім принципом вже нову групу вершин, але індексуємо дану групу як i=i+1 та s= s+1;

3) повторюємо операцію (2) доти, поки n > 0, тобто поки не будуть помічені всі вершини графу.

В глобальну пам'ять відеокарти мають бути скопійовані наступні дані:

- кількість вершин графа N;
- одномірний масив об'єктів вершин графа $V = \{v_i\}, i = \overline{1,N}$. У даному випадку вершинами є об'єкти, що моделюють поведінку кожної елементарної системи нечіткого логічного виведення Такагі – Сугено;
- кількість ярусів графа s;
- одномірний масив значень ширини кожного ярусу $L = \{l_i\}, i = \overline{1,s}$;
- одномірний масив номерів вершин у тому порядку, в якому вони були розділені на групи $M = \{m_i\}, i = \overline{1,N}$, як показано на рис. 2.

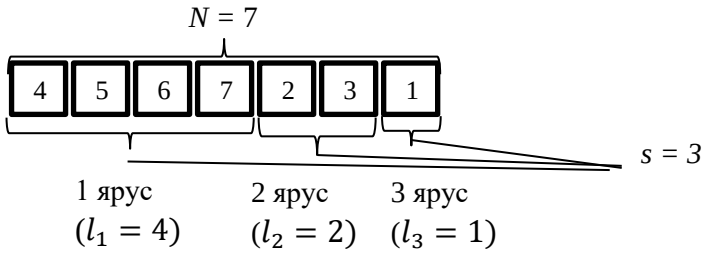

Рис. 2. Масив послідовності номерів вершин у порядку слідування груп

Індексація вершин графу. Для копіювання вхідних даних в пам'ять GPU, об'єкти-вершини ярусно-паралельної схеми зберігаються в одномірному масиві. Тому виникає проблема індексації вершин з урахуванням номеру яруса. Далі представлено метод індексації вершин у масивах, в яких зберігаються графи залежностей блоків правил нечітких систем та, відповідно, спосіб визначення індексів вершин кожного ярусу.

Нехай N – кількість вершин графа (кількість блоків правил в ярусі), s – кількість ярусів. Тоді, якщо кількість систем в i-му ярусі – l_i, $i = \overline{1,s}$, то маємо наступне співвідношення:

$$\sum_{i=1}^{s} l_i = N .$$

Нехай V – масив вершин, M – масив номерів вершин. Визначимо для кожного ярусу i групу вершин $D_i = \left\{d_p^{(i)}\right\}$, $p = \overline{1,l_i}$ у масиві V. Тоді кожна вершина i-го ярусу в масиві V індексується за наступною формулою:

$$d_p^{(i)} = M_{z_p^{(i)}}, \; z_p^{(i)} = p + \sum_{k=1}^{i} l_k,$$

де i – номер ярусу, p – номер вершини в i-му ярусі, $z_p^{(i)}$ – індекс номера вершини p i-го ярусу в масиві M. $M_{z_p^{(i)}}$ – це обчислений індекс вершини $d_p^{(i)}$ в масиві V.

Внутрішній паралелізм. Кожна вершина (елементарна нечітка система) містить множину об'єктів для здійснення нечіткого виведення за певною множиною вхідних значень. Особливістю обчислення кожного правила є можливість їх активізації у будь-якій послідовності, тобто вихідні значення правил можуть обчислюватися незалежно одне від одного. Таким чином, обчислення значень за всіма правилами в межах кожної елементарної нечіткої системи (кожної вершини) можна виконувати паралельно. В цьому випадку кожна вершина v_i являє собою окремий підграф, а кожне правило можна розглядати як окрему вершину підграфа v_i. Нехай кожна v_i вершина графа G – орієнтований ациклічний мультиграф $R = \{H, Y\}$, де H – множина вершин, а Y – множина ребер графа R. $H = \{h_j\}, j = \overline{1, M_i + 1}$, де M_i – кількість правил i-ї вершини графа G, а $M_i + 1$ вершиною є функція нечіткого логічного виводу. $Y = \{y_d\}$, $d = \overline{1, M_i}$, де M_i – кількість правил i-ї вершини графа G. У кожному графі R від кожної вершини $h_j \ldots h_{M_i}$ проведені дуги до вершини h_{M_i+1} (рис. 3). Такий граф має назву канонічної паралельної форми [10] і відповідно значення правил у вершинах $h_j \ldots h_{M_i}$ можуть бути обчислені паралельно в межах одного ярусу.

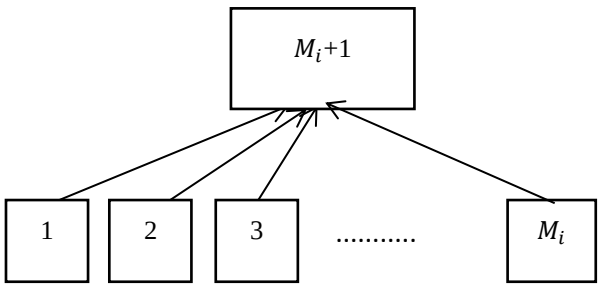

Рис. 3. Представлення графа R

Паралельне виведення в ієрархічних нечітких системах. Найважливішим етапом є здійснення обчислень результуючих значень ієрархічної системи на графічних прискорювачах Nvidia. Графічний процесор складається з сітки Grid, яка поділяється на блоки (blocks). Кожний блок, у свою чергу містить нитки (threads). Розподіл обчислень в ієрархічних нечітких системах між блоками та нитками показано на рис. 4.

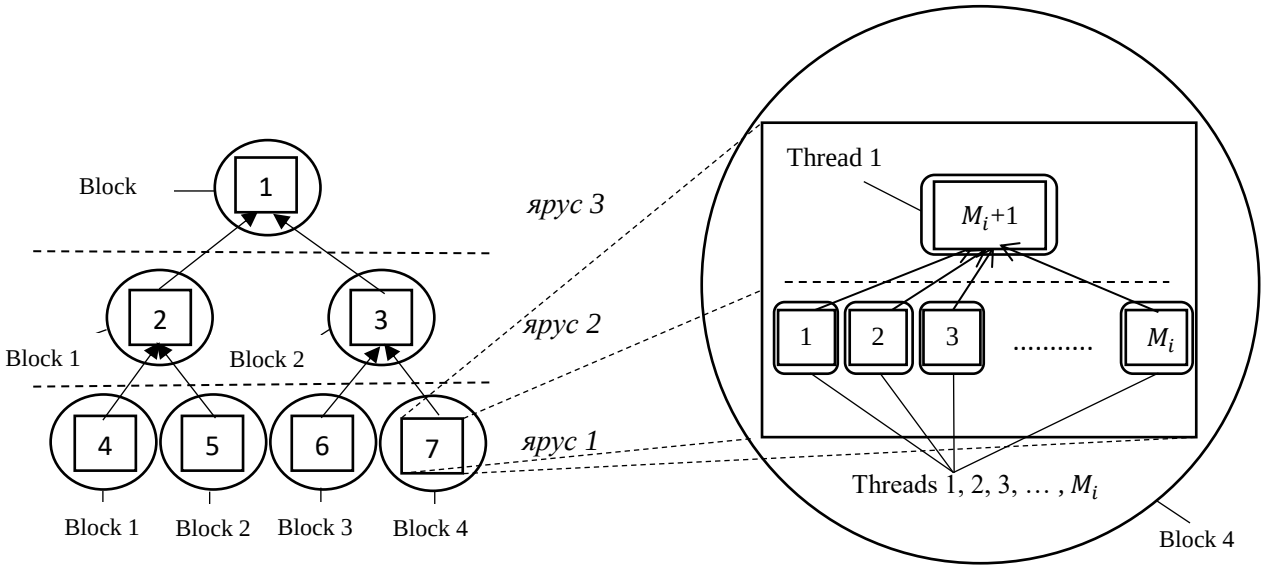

Рис. 4. Схема розподілу обчислень в ієрархічних нечітких системах на основі GPU

Основна ідея застосування технології CUDA при побудові ієрархічних систем нечіткого логічного виведення полягає у тому, що результуюче значення кожної елементарної нечіткої системи (кожної вершини v_i графа G) обчислюється окремим блоком правил, у якому, в свою чергу, обробка кожного правила (кожної вершини h_j графа R) системи розподіляється за нитками зазначеного блоку.

Таким чином, для досягнення більшого прискорення логічного виведення в ієрархічних нечітких системах Такагі – Сугено ефективно реалізовано як паралельне обчислення результатів блоків правил, так і можливість внутрішнього паралелізму за правилами в кожному окремому блоці (рис. 4).

Відеокарта архітектури 6.1. Pascal, яка використовувалась для експериментальних обчислень побудованої програмної системи, має 768 процесорних ядер, що обчислюють виділені блоки та нитки в них, що дозволяє ефективно реалізувати внутрішній паралелізм при нечіткому виведенні. Внутрішній паралелізм, у свою чергу, мінімізує втрати процесорних ресурсів, максимально навантажуючи ядра графічного процесора.

Залучення великої кількості процесорів на CPU вимагає залучення суперкомп'ютера, проте має суттєві обмеження, що пов'язані з відсутністю спільної пам'яті та додатковими витратами часу на обмін даними між процесорами розподіленої системи.

Оцінка прискорення нечіткого логічного виведення в ієрархічних системах. При використанні графічних прискорювачів досягається значне прискорення обчислень для ієрархічних нечітких систем, яке набагато більше ніж для паралельних обчислень таких саме систем на основі MPI (рис. 5). Такий результат отримано саме за рахунок масового паралелізму, використання загальної пам'яті, а особливо за рахунок розпаралелювання всередині кожної системи за правилами, тобто внутрішнього паралелізму. Зазначений результат експериментально підтверджено для ієрархічних нечітких систем, залежності входів-виходів між окремими блоками правил якого задаються графом великої розмірності (N=1000 блоків) та генеруються випадковим чином. Кількість ребер q задається як $q = 4 * n$, де n – кількість згенерованих вершин графа.

Очевидно, що зі збільшенням кількості систем збільшується і прискорення (рис. 5). Для ієрархічних нечітких систем, що складаються із 1000 елементарних систем, прискорення на основі технології CUDA – більше ніж у 500 разів у порівнянні з послідовним варіантом. Варіант програми на базі MPI на 24 процесорах прискорює нечітке виведення не більше, ніж у 20 раз, але в 25 разів менше за GPU.

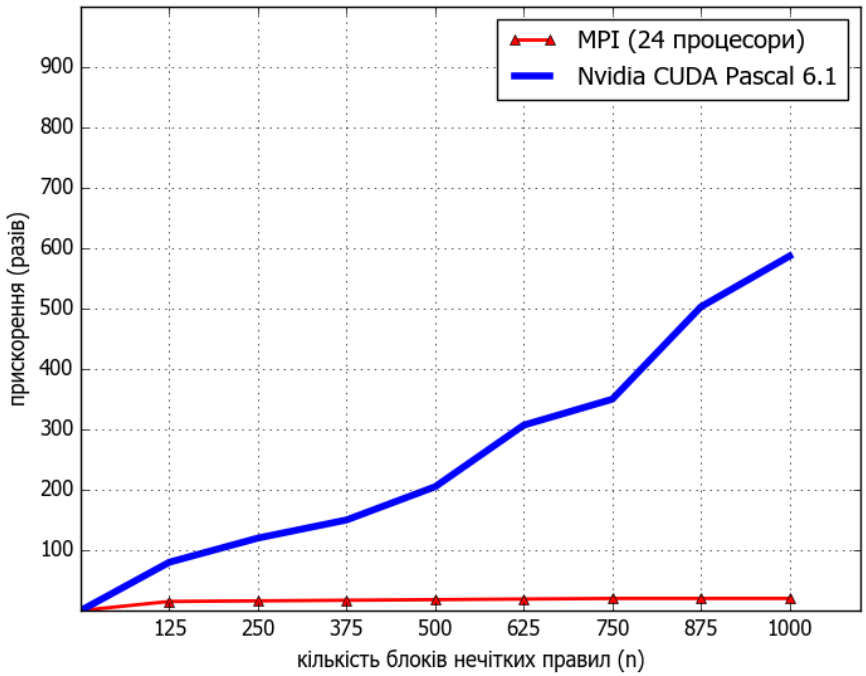

Рис. 5. Прискорення нечіткого виведення для ієрархічних нечітких систем

Побудова інтелектуальної системи оцінки привабливості стартапів на основі ієрархічної нечіткої системи

Метою даної інтелектуальної системи є аналіз та оцінювання якості стартапів за різними показниками та отримання комплексної оцінки привабливості даних стартапів. Під привабливістю стартапу часто розуміють доцільність його подальшого запуску, фінансування та роботи над ним [13].

На сьогоднішній день вищезазначена задача є дуже актуальною. Невеликі інноваційні проекти все більше займають місце в малому бізнесі, проте виникає необхідність попередньої оцінки проекту на предмет прибутковості та зменшення ризиків для потенційних інвесторів та розробників таких проектів.

Базу знань у вигляді нечітких правил розробляє експерт або група експертів з даної предметної області. Програмна система ґрунтується на лінгвістичних правилах, таких як, *if x is high then y is good.* Спеціальний програмний модуль системи обробляє лінгвістичні вирази, переводить їх до представлення у вигляді нечітких множин.

Всього ієрархічна нечітка система оцінки привабливості стартапів містить 162 правила. Враховуючи, що кожна з 15 вхідних змінних може бути представлена одним з трьох лінгвістичних значень, побудова відповідної повної бази для системи без ієрархії вимагає створення 3^{15} нечітких правил. Очевидно, що побудувати та перевірити таку кількість правил для аналогічної, але однорівневої системи не під силу навіть досвідченим експертам.

Зазначимо, що розробка ієрархічних систем нечіткого логічного виведення дозволяє значно збільшити кількість факторів, що впливають на оцінку привабливості стартапів, дозволяючи прийняти більш об’єктивне рішення щодо подальшого фінансування. При цьому кількість правил, які має написати експерт, за рахунок декомпозиції, значно зменшується. Граф залежностей між блоками нечітких правил, їх вхідні та вихідні змінні показано на рис. 6.

На рис. 7 показано лінгвістичні терми (Високий, Середній, Низький) та відповідні функції належності, які використовуються в інтелектуальній системі оцінки привабливості стартапів. Використані трикутні функції належності на інтервалі можливих значень від одного до п’яти (рис. 7).

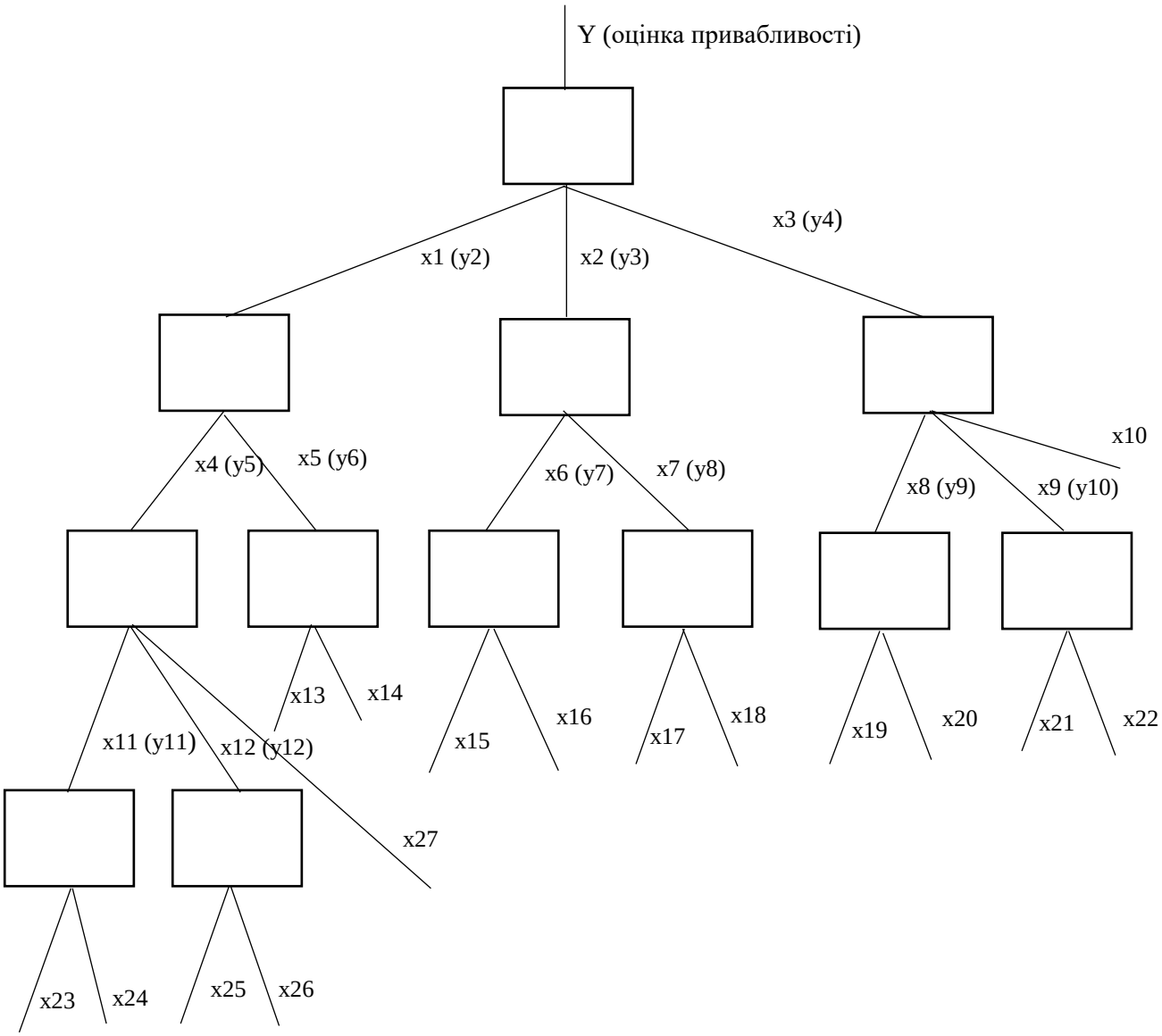

Рис. 6. Граф залежностей між блоками нечітких правил в інтелектуальній системі оцінки привабливості стартапів

Для побудови нечіткої ієрархічної системи оцінки привабливості необхідно визначити простір вхідних факторів $X = \{x_i\}$, $i = \overline{1,n}$ (табл. 1), та простір вихідних значень системи $Y = \{y_j\}$, $j = \overline{1,m}$ (табл. 2), що являють собою різні показники проекту. На основі критеріїв, що представлені в табл. 1, можна визначити вхідні змінні, що необхідні для нечіткого виведення в інтелектуальній системі.

Представлена система є ієрархічною (рис. 6), тобто всі вхідні змінні задаються користувачем до початку нечіткого виведення, а інші змінні обчислюються з попередньо визначених змінних нечітких блоків правил від яких вони залежать. Для здійснення нечіткого виведення користувачу необхідно задати наступні фактори: *x23* – стійкість проектних рішень (чи змінюються рішення під впливом зовнішніх факторів); *x24* – менеджмент проекту (якість керування проектом); *x25* – можливість виводу коштів у разі провалу проекту або за власним бажанням; *x26* – технічний ризик; *x13* – стартовий капітал (наявність власних коштів для започаткування справи); *x14* – подальше власне фінансування (чи є можливість у подальшому підтримувати стартап власними коштами); *x15* – досвід керівника за тематикою подібних проектів; *x16* – рівень освіти керівника; *x17* – готовність команди працювати повний робочий день (суміс-

ники або штатні працівники); *x18* – якість колективної праці (результативність роботи); *x19* – масштаб інновацій (рівень визнання інновацій); *x20* – тип інновацій (оригінальність проектних рішень); *x21* – пріоритет ринку (обмеження за типом споживачів продукції стартапу); *x22* – розмір ринку (обмеження за кількістю споживачів продукції стартапу); *x10* – ступінь розробки (на якій стадії знаходиться розробка даного проекту). Результуюча змінна *y* є остаточною оцінкою рейтингу стартапу на основі результатів обчислень підсистем нижчих рівнів та кінцевої обробки цих даних. Змінна *y* може приймати такі лінгвістичні значення: схвалити стартап, схвалити після доробки або відхилити.

Таблиця 1

Специфікація вхідних змінних інтелектуальної системи (фрагмент)

Змінна	**Найменування фактору**	**Значення**	**Зміст**
x1	Фінансування	Низьке (Н)	Недостатнє фінансування для даного проекту
		Середнє (С)	Фінансування проекту пов'язане з деякими труднощами
		Високе (В)	Фінансування повністю покриває вартість проекту
x2	Кадри	Низьке (Н)	Кадри є профнепридатними
		Середнє (С)	Потрібна додаткова професійна підготовка
		Високе (В)	Кадри повністю задовольняють вимогам проекту
x3	Проект	Низький (Н)	Проект приречений на провал, вимагає повного перегляду
		Середній (С)	Має слабкі сторони, вимагає деякої переробки
		Високий (В)	Проект може зайняти гідне місце на ринку
x4	Інвестиції	Низькі (Н)	Відсутні перспективи інвестування
		Середні (С)	Деякі інвестори готові почати співпрацювати
		Високі (В)	Присутні один або більше інвесторів, з перспективою подальшого капіталовкладення
…	…	…	…
x24	Менеджмент проекту	Низький (Н)	Слабке планування або відсутність планування і моніторингу
		Середній (С)	Планування і моніторинг з виконання завдань
		Високий (В)	Планування системи і моніторинг процесів проводяться своєчасно
x25	Можливість виводу коштів	Низька (Н)	Виведення коштів з проекту неможливий
		Середня (С)	Виведення коштів можливий тільки з певним відсотком втрат
		Висока (В)	Можливість повного виведення коштів
x26	Технічний ризик	Низький (Н)	Вимоги до проекту відсутні
		Середній (С)	Вимоги до проекту сформульовані нечітко
		Високий (В)	Вимоги до проекту чітко сформульовані

Таблиця 2

Специфікація результуючих та проміжних змінних інтелектуальної системи (фрагмент)

Змінна	**Найменування фактору**	**Значення**	**Зміст**
Y	Оцінка привабливості	Низька (Н)	Не варто займатися даним проектом
		Середня (С)	Проект має ризик провалу
		Висока (В)	Проектом варто займатися
y2	Фінансування	Низьке (Н)	Не варто займатися даним проектом
		Середне (С)	Проект має ризик провалу
		Високе (В)	Проектом варто займатися
…	…	…	…
y11	Ефективність інвестицій	Низька (Н)	Потенційний прибуток інвестора 10 % в рік
		Середня (С)	Потенційний прибуток інвестора 50 % в рік
		Висока (В)	Потенційний прибуток інвестора 100 % в рік
y12	Безпека	Низька (Н)	90 і більше відсотків, що інвестор втратить капітал
		Середня (С)	До 60 відсотків ймовірності збереження капіталу
		Висока (В)	Імовірність збереження капіталу понад 95 відсотків

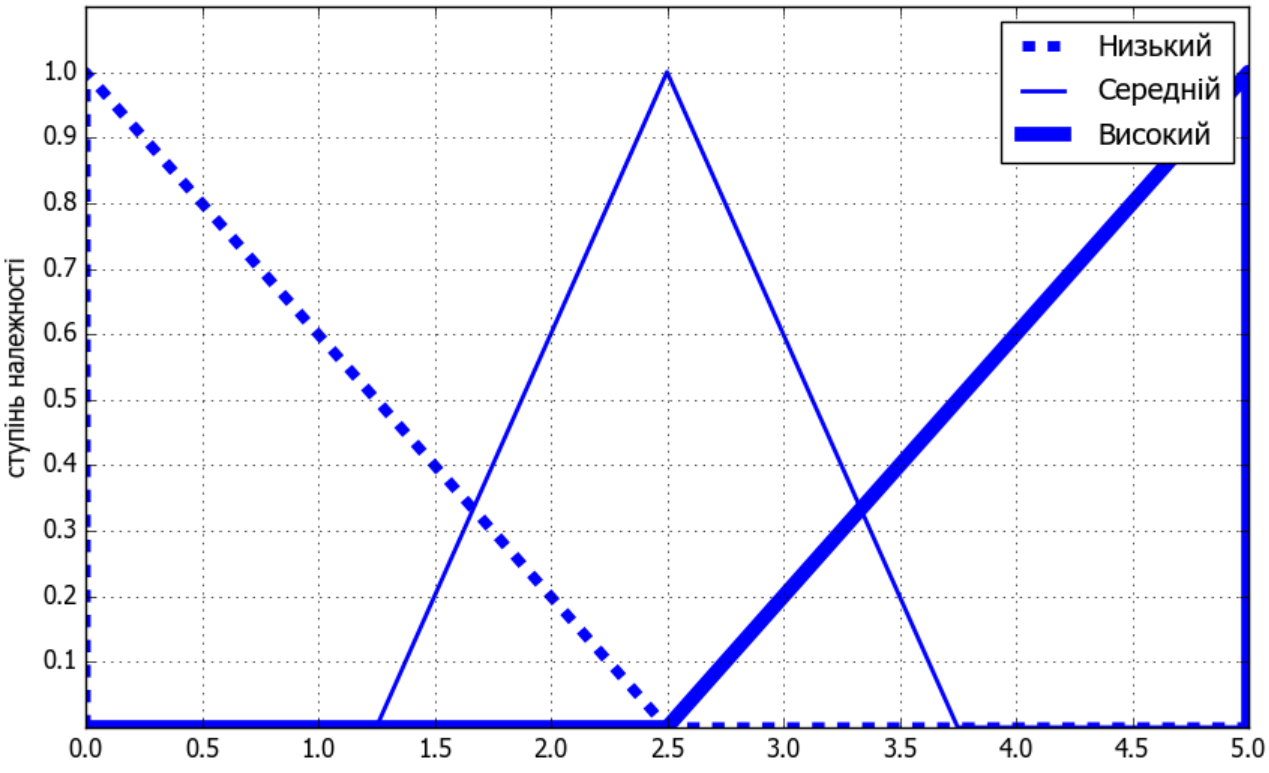

Рис. 7. Графічне представлення функцій належності

В табл. 3 представлено отримані рішення інтелектуальної системи для різних факторів у п'яти різних випадках. Очевидно, що проекти стартапів № 1–3 рекомендовано відправити на доробку, стартап № 5 визнаний привабливим для інвестування, а стартап № 4 вимагає повного перегляду та для інвестування не рекомендований.

Зазначимо, що така ієрархічна інтелектуальна система паралельно обчислює результуючі значення вершин графа залежностей (елементарних нечітких систем) в межах кожного ярусу. Завдяки внутрішньому паралелізму обробка всіх правил кожної вершини виконується також паралельно, для досягнення більш високих показників прискорення. Наприклад, вершина графу залежностей (блок правил), що розраховує остаточну оцінку привабливості стартапу, містить наступні правила:

if x1 is low and x2 is average and x3 is low then y1 is low

if x1 is low and x2 is average and x3 is average then y1 is average

……..

if x1 is average and x2 is high and x3 is high then y1 is high,

де *low, average, high* відповідають значенням *Низький, Середній, Високий* відповідно. Обробку кожного правила в межах всього блоку можна проводити незалежно одне від одного. Саме за рахунок цього кожний блок правил виконується паралельно, тобто обчислення кожного правила розпочинається одночасно окремою ниткою (thread) графічного процесору відеокарти (рис. 4).

Прискорення нечіткого логічного виведення для інтелектуальної системи оцінки привабливості стартапів на основі технологій MPI та CUDA у порівнянні з однопроцесорним варіантом такої системи, представлено на рис. 8. Зазначимо, що технологія CUDA дає прискорення приблизно в 100 разів порівняно з послідовним варіантом, а MPI дає прискорення, для цієї системи нечіткого виведення тільки в 3 рази. Для ієрархічної нечіткої системи з 12 блоками правил технологія CUDA дає прискорення більше, ніж в 30 разів, порівняно з технологією MPI на однакових вхідних даних. Це пов'язано, в першу чергу, з використанням внутрішнього паралелізму.

Таблиця 3

Результати роботи інтелектуальної системи за різних вхідних даних

Стартап №	Вхідні фактори стартапів для оцінки привабливості																**Y**	
	Порядкові номери вхідних змінних ієрархічної системи (**X**)																	
	23	***24***	***25***	***26***	***13***	***14***	***15***	***16***	***17***	***18***	***19***	***20***	***21***	***22***	***10***	***27***		
1	3	2	4	5	1	3	2	3	5	2	2	1	2	4	3	5	**3,0001**	**C**
2	3	4	3	1	4	2	2	3	2	3	2	5	4	2	3	3	**3,0007**	**C**
3	3	4	4	4	3	4	2	2	4	5	5	4	3	4	3	4	**3,0005**	**C**
4	1	2	2	1	3	3	3	3	2	1	4	3	1	1	1	2	**1,5986**	**H**
5	4	5	5	5	5	4	3	4	5	5	5	5	4	5	4	5	**4,7320**	**B**

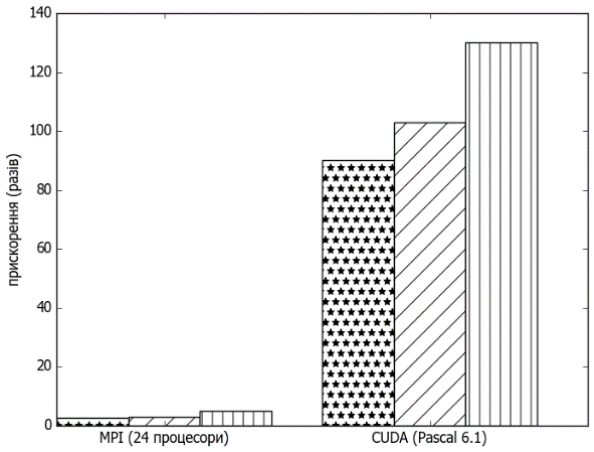

Рис. 8. Прискорення нечіткого виведення для MPI та CUDA

Висновки

В роботі розглянуто метод розпаралелювання систем нечіткого логічного виведення на основі багатопотокового програмування графічних прискорювачів Nvidia CUDA. Обґрунтовано переваги даного методу, встановлено оцінки прискорення систем нечіткого виведення, що побудовані за зазначеним методом, здійснено порівняльну характеристику з варіантами нечітких систем, що реалізовані на базі технології паралельного програмування MPI. В роботах [3,4] для побудови нечітких систем застосовувалася технологія паралельного програмування MPI, що має такі переваги, як відносна простота використання та висока переносимість коду. Дослідження та експерименти, проте, виявили ряд обмежень технології MPI при розпаралелюванні ієрархічних систем нечіткого логічного виведення. Зокрема, це необхідність налагоджувати механізми передачі даних між процесами, що ускладює код програми та уповільнює роботу програми за наявності великих обсягів даних, що передаються. Суттєвим є обмеженість процесорними ресурсами, які водночас є досить дорогими, оскільки для отримання кількох десятків паралельних процесів потрібно задіювати суперкомп'ютер або розподілену комп'ютерну мережу. Для подолання зазначених обмежень авторами розроблено та обгрунтовано метод нечіткого логічного виведення на основі багатопотокового програмування Nvidia CUDA. Застосування апаратної архітектури останнього покоління Pascal 6.1 надає у розпорядження досить велику кількість потоків для паралельної обробки нечітких правил, що повністю задовольняє потреби при нечіткому виведенні для систем дуже великої розмірності (кількість вхідних та вихідних змінних). Однією з головних переваг розробленого метода на основі графічних прискорювачів CUDA є можливість застосувати внутрішнє розпаралелювання за правилами для кожного блоку нечітких правил, що неможливо ефективно здійснити з використанням технології MPI через необхідність створення та координації великої кількості потоків. Це дозволяє отримати вагомі оцінки прискорення нечіткого логічного виведення у сотні разів (для окремих графів залежностей) у порівнянні з MPI.

Розроблено програмну систему оцінки привабливості стартапів на основі ієрархічних систем нечіткого логічного виведення Такагі – Сугено. На базі ярусно-паралельної форми обчислень та з використанням технології CUDA 8.0 в середовищі програмування Visual Studio 2012 побудовано програмну систему для оцінки привабливості стартапів.

1. Парасюк И.Н., Ершов С.В. Мультиагентные модели на основе нечеткой логики высшего типа для высокопроизводительной среды. *Проблеми програмування*. 2012. № 2-3. С. 260–269.
2. Штовба С.Д. Проектирование нечетких систем средствами MATLAB. М.: Горячая линия. Телеком. 2007. 288 с.
3. Torra V. A. review of the construction of hierarchical fuzzy systems. Int. J. Intell. Syst. 2002. N 17. P. 531–543.
4. Yager R.R. On a hierarchical structure for fuzzy modeling and control. IEEE Trans. Syst. Man Cybern. 1993. N 23. P. 1189–1197.
5. Yager R.R. On the construction of hierarchical fuzzy systems models. IEEE Trans. Syst. Man Cybern. 1998. N 28. P. 55–66.
6. Cordon O., Herrera F., Zwir I. Linguistic modeling by hierarchical systems of linguistic rules. IEEE Trans. Fuzzy Syst. 2002. N 10. P. 2–20.

7. Raju G.V.S., Zhou J., Kisner R.A. Hierarchical fuzzy control. Int. J. Control. 1991. N 54. P. 1201–1216.
8. Єршов С.В., Пономаренко Р.М. Паралельні моделі багаторівневих нечітких систем Такагі – Сугено. *Проблеми програмування.* 2016. № 1. С. 141–149.
9. Єршов С.В., Пономаренко Р.М. Ярусно-паралельна модель обчислень для логічного виведення у нечітких багаторівневих системах. Комп'ютерна математика. 2016. № 1. С. 28–36.
10. Воеводин В.В. Параллельные вычисления. СПб: БХВ-Петербург. 2002. 608 с.
11. http://www.nvidia.com.ua/object/cuda-parallel-computing-ru.html
12. Дорошенко А.Ю., Бекетов О.Г. Метод паралелізації циклів сіткових обчислювальних задач для графічних прискорювачів. *Проблеми програмування.* 2017. № 1. С. 59–66.
13. Киселева Е.М., Притоманова О.М., Журавель С.В. Оценка инвестиционной привлекательности стартапов на основе нейронечетких технологий. *Проблемы управления и информатики*. 2016. № 5. С. 123–143.

References

1. Parasyk I.N, Yershov S.V. (2012) Multilevel models base on fuzzy logic of the highest type for high-performance environment // Problems in programming. – N 2-3. P. 260–269.
2. Shtovba S.D. (2007) Fuzzy systems design by MATLAB mean // M.: Goryachaya liniya – Telecom – 288 p.
3. Torra V. (2002) A review of the construction of hierarchical fuzzy systems // Int. J. Intell. Syst. – Vol.17, N 5. – P. 531–543 .
4. Yager R.R. (1993) On a hierarchical structure for fuzzy modeling and control // IEEE Trans. Syst. Man Cybern. – Vol. 23, N 4. – P. 1189–1197.
5. Yager R.R. (1998) On the construction of hierarchical fuzzy systems models // IEEE Trans. Syst. Man Cybern. – Vol. 28, N 1. – P. 55–66.
6. Cordon O., Herrera F., Zwir I. (2002) Linguistic modeling by hierarchical systems of linguistic rules // IEEE Trans. Fuzzy Syst. – N 10. – P. 2–20.
7. Raju G.V.S., Zhou J., Kisner R.A. (1991) Hierarchical fuzzy control // Int. J. Control. – Vol. 54, N 5. – P. 1201–1216.
8. Yershov S.V., Ponomarenko R.M. (2016) Parallel models for Takagi-Sugeno's fuzzy multilevel systems // Problems in programming. – N 1. – P. 141–149.
9. Yershov S.V., Ponomarenko R.M. (2016) Tier parallel computing model for logical inference on fuzzy multilevel systems // Mathematic for computers. – N 1. – P. 28–36.
10. Voevodin V.V. (2002) Parallel inference // SPb: BHV-Peterburg. – 608 p.
11. Nvidia CUDA. – http://www.nvidia.com.ua/object/cuda-parallel-computing-ru.html.
12. Doroshenko A.U., Beketov O.G. (2017) The cycle paralellization method of the mesh tasks for graphic accelerator // Problems in programming. – N 1. – P. 59–66.
13. Kiseleva E.M., Prytomanova O.M., Zhuravel S.V. (2016) Evaluation of the start-ups investment attractiveness base on the neuro-fuzzy technologies // Problems in management and informatics. – N 5. – P. 123–143.

Одержано 04.10.2017

Про авторів:

Єршов Сергій Володимирович,
доктор фізико-математичних наук,
завідувач відділу інтелектуальних програмних систем Інституту кібернетики імені В.М. Глушкова НАН України.
Кількість наукових публікацій в українських виданнях – 67.
Кількість наукових публікацій в зарубіжних виданнях – 9.
http://orcid.org/0000-0002-9895-777X

Пономаренко Роман Миколайович,
аспірант Інституту кібернетики імені В.М. Глушкова НАН України,
Кількість наукових публікацій в українських виданнях – 3.
http://orcid.org/0000-0001-9681-2297

Місце роботи авторів:

Інститут кібернетики
імені В.М. Глушкова НАН України,
проспект Академіка Глушкова, 40,
Київ, 03187, Україна.
Тел.: (044) 526 41 78.
E-mail: sershv@ukr.net.

Тел.: (044) 526 64 22.
E-mail: ponomarenko_roman@ukr.net