УДК 004.94:004.415

М. П. Дивак, д-р техн. наук, проф.; О. В. Кіндзерський

АРХІТЕКТУРА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ СТРУКТУРНОЇ ТА ПАРАМЕТРИЧНОЇ ІДЕНТИФІКАЦІЇ НА ОСНОВІ АЛГОРИТМУ ШТУЧНОЇ БДЖОЛИНОЇ КОЛОНІЇ З ВИКОРИСТАННЯМ ТЕХНОЛОГІЇ NVIDIA CUDA

В статті представлено комплексну архітектурну концепцію та реалізацію програмного забезпечення структурної та параметричної ідентифікації на основі алгоритму Штучної Бджолиної Колонії (ШБК) з використанням технології Nvidia CUDA. Система інтегрує три різні технологічні шари: користувацький інтерфейс Windows Forms, що забезпечує інтуїтивну конфігурацію параметрів та можливості візуалізації в реальному часі, високопродуктивну C# бібліотеку, що реалізує основний алгоритм ШБК з підтримкою як параметричних, так і структурних підходів до оптимізації, та Nvidia CUDA бекенд. Архітектура демонструє виняткову масштабованість, підтримуючи множинні стратегії виконання, включаючи послідовну обробку для маломасштабних проблем, паралельні обчислення CPU для помірнорозмірних оптимізацій та GPU-прискорені обчислення для великомасштабних, багатовимірних проблем. Модульний дизайн системи включає сучасні принципи інженерії програмного забезпечення, що характеризуються чітким розділенням відповідальності, комплексними механізмами обробки помилок та логування, розширюваними інтерфейсами, що сприяють інтеграції нових алгоритмів оптимізації та методів оцінки. Ключові архітектурні інновації включають динамічну генерацію CUDA-ядер, що адаптується до специфічних характеристик проблем, підтримку інтервальної арифметики для надійної оптимізації в умовах невизначеності та складну систему управління даними, що обробляє багатовимірні простори проблем з ефективним управлінням пам'яттю та очищенням ресурсів. Реалізація підтримує збереження проєктів у форматі JSON, що забезпечує відтворюваність досліджень та співпрацю в оптимізації, одночасно надаючи комплексні можливості імпорту/експорту даних у форматах CSV та JSON для безшовної інтеграції з зовнішніми інструментами та наборами даних. Аналіз продуктивності демонструє значні прискорення для великомасштабних проблем оптимізації через GPU-прискорення, зберігаючи при цьому гнучкість для обробки різноманітних предметних областей від оптимізації математичних функцій до складних інженерних проблем проєктування. Розширюваність архітектури демонструється через підтримку користувацьких функцій мети через динамічну компіляцію з використанням Roslyn-скриптингу, що дозволяє користувачам визначати специфічні для проблеми критерії оцінки як виконуваний C# код. Надійність системи забезпечується через комплексну перевірку вхідних даних, механізми обробки винятків та детальний моніторинг продуктивності, що допомагає в налагодженні та налаштуванні оптимізації. Цей архітектурний підхід надає дослідникам та інженерам потужну, масштабовану та підтримувану платформу для вирішення складних проблем оптимізації в багатьох вимірах, що робить її придатною як для академічних дослідницьких застосувань, так і для промислових викликів оптимізації, одночасно встановлюючи основу для майбутніх покращень, включаючи можливості розподілених обчислень.

Ключові слова: *алгоритм штучної бджолиної колонії, параметрична ідентифікація, структурна ідентифікація, паралельні обчислення, Nvidia CUDA.*

Вступ

Сучасні наукові та інженерні дослідження часто стикаються з проблемою ідентифікації математичних моделей, що найкраще описують спостережувані явища або процеси. Ця проблема має два основні аспекти: структурну ідентифікацію та параметричну ідентифікацію, кожна з яких представляє унікальні виклики та вимагає спеціалізованих підходів до оптимізації.

Структурна ідентифікація полягає у визначенні математичної структури моделі, що найкраще відповідає експериментальним даним. Це включає вибір типу функціональної

залежності (лінійна, поліноміальна, експоненціальна, логарифмічна тощо), визначення кількості та характеру змінних, а також встановлення зв'язків між ними. Структурна ідентифікація є особливо складною, оскільки простір можливих математичних структур є дискретним, необмеженим та часто нелінійним. Традиційні методи оптимізації, засновані на градієнтних підходах, часто неспроможні ефективно досліджувати такий простір через наявність локальних мінімумів та розривів у функції придатності.

Параметрична ідентифікація фокусується на налаштуванні числових параметрів вже визначеної математичної структури для мінімізації різниці між прогнозованими та спостережуваними значеннями. Хоча ця проблема є більш структурованою порівняно зі структурною ідентифікацією, вона все ще може бути складною через наявність нелінійних залежностей, взаємодії між параметрами та шуму у експериментальних даних. Крім того, багатопараметричні моделі можуть мати високий рівень кореляції між параметрами, що призводить до проблем ідентифікації та нестабільності рішень.

Метаевристичні алгоритми, особливо ті, що засновані на ройовому інтелекті, показали себе як ефективні інструменти для вирішення проблем структурної та параметричної ідентифікації. Штучна Бджолина Колонія (ШБК) алгоритм, запропонований Карабогою [1], є особливо перспективним через його здатність ефективно досліджувати складні простори рішень та уникати локальних мінімумів. Алгоритм імітує поведінку бджолиної колонії в пошуку нектару, де різні типи бджіл (працюючі, спостерігачі та розвідники) виконують різні ролі в процесі оптимізації.

Сучасні проблеми ідентифікації часто вимагають обробки великих обсягів даних та виконання тисяч або мільйонів оцінок придатності. Це створює значні виклики з точки зору обчислювальної продуктивності, особливо коли розглядаються як структурні, так і параметричні аспекти ідентифікації. GPU-прискорені обчислення через Nvidia CUDA надають можливість значного прискорення цих обчислень, дозволяючи обробляти великомасштабні проблеми ідентифікації в розумні часові рамки.

Програмне забезпечення оптимізації Штучної Бджолиної Колонії представляє багатошарову архітектуру, розроблену для вирішення складних проблем оптимізації через реалізацію алгоритмів ройового інтелекту [2 – 8]. Система складається з трьох основних компонентів: користувацького інтерфейсу, основної бібліотеки оптимізації та CUDA-прискореного бекенду для високопродуктивних обчислень [8]. Це розділення відповідальностей забезпечує підтримуваність, розширюваність та ефективне використання ресурсів [9, 10].

Постановка задачі та алгоритм її розв'язування

Розглянемо задачу структурної ідентифікації інтервальних моделей динамічних об'єктів. Математична модель об'єкта представлена у вигляді різницевого рівняння:

$$v_k = \vec{f}^T(v_{k-d}, \dots, v_{k-1}) * \vec{g}, k = d, \dots, K \quad (1)$$

де v_k – модельована характеристика динамічного об'єкта на часовій дискреті $k=d,,K$, d – порядок дискретної моделі, $\vec{g}$ – вектор невідомих параметрів моделі, $\vec{f}^T$ – вектор базисних функцій.

На відміну від задачі параметричної ідентифікації, вектор базисних функцій в задачі структурної ідентифікації є невідомим.

Для представлення математичної моделі характеристики динамічного об'єкта уведемо поняття множини структурних елементів:

$$\lambda_s = \{f_1^s(v_{k-d}, \dots, v_{k-1}) * g_1^s, \dots, f_m^s(v_{k-d}, \dots, v_{k-1}) * g_m^s\} \quad (2)$$

У виразі s – означає певний набір структурних елементів, на основі якого будуємо s-ту модель у вигляді (1), тоді λ_s використовуємо для позначення s-ої структури.

Математичні моделі, які розглядатимемо у процесі структурної ідентифікації будуть мати наступний вигляд:

$$v_k(\lambda_s) = f_1^s(v_{k-d}, \dots, v_{k-1}) * g_1^s + \dots + f_m^s(v_{k-d} \dots, v_{k-1}) * g_m^s \quad (3)$$

Зважаючи на отримані експериментальні інтервальні дані задаємо умову узгодженості математичної моделі:

$$v_k(\lambda_s, V_k) \in [z_k^-, z_k^+], \forall k = 0, \dots, d-1, d, \dots, K \quad (4)$$

де z_k^-, z_k^+ – нижня і верхня межа експериментально отриманих значень характеристики, $v_k(\lambda_s, V_k)$ – означає істинне значення вихідної характеристики для фіксованого набору структурних елементів λ_s і для фіксованих значень вектора $\vec{V} = (v_{k-d}, \dots, v_{k-1})^T$, у часових дискретах $k=0,\dots,K$.

Приймаючи до уваги умову (4) отримаємо наступну систему:

$$\begin{cases} z_0^- \le f_1^s(\vec{V}_0) * g_1^s + \dots + f_m^s(\vec{V}_0) * g_m^s \le z_0^+ \\ \vdots \\ z_{d-1}^- \le f_1^s(\vec{V}_{d-1}) * g_1^s + \dots + f_m^s(\vec{V}_{d-1}) * g_m^s \le z_{d-1}^+ \\ z_d^- \le f_1^s(\vec{V}_d) * g_1^s + \dots + f_m^s(\vec{V}_d) * g_m^s \le z_d^+ \\ \vdots \\ z_K^- \le f_1^s(\vec{V}_K) * g_1^s + \dots + f_m^s(\vec{V}_K) * g_m^s \le z_K^+ \end{cases} \quad (5)$$

Спираючись на той факт що в отриманій ІСНАР перші d інтервальних рівнянь це початкові умови, то перепишемо її у наступному вигляді

$$\begin{cases} [v_0^-; v_0^+] \subseteq [z_0^-, z_0^+], \dots, [v_{d-1}^-; v_{d-1}^+] \subseteq [z_{d-1}^-, z_{d-1}^+] \\ \vdots \\ z_k^- \le f_1^s(\vec{V}_k) * g_1^s + \dots + f_m^s(\vec{V}_k) * g_m^s \le z_k^+, k = d, \dots, K \end{cases} \quad (6)$$

Після проведення даних перетворень ми отримали загальну форму задачі параметричної ідентифікації інтервальних моделей динамічних об'єктів у вигляді ІНСАР для окремої s-ої моделі претендента. Розв'язок цієї системи отримуємо з реалізації ітераційної процедури, на кожній ітерації якої, обчислюємо функцію якості оцінки параметрів математичної моделі. У результаті перетворень, детально описаних в [4], задачу структурної ідентифікації інтервальних моделей динамічних об'єктів сформуємо у вигляді оптимізаційної задачі:

$$(\lambda_s) \xrightarrow{\lambda_s = \left\{ f_1^s(\vec{V}) * g_{l1}^s, \dots, f_{m_s}^s(\vec{V}) * g_{lm_s}^s \right\}} min, \quad (7)$$

$$(m_s \in [I_{min}, I_{max}], f_1^s(\vec{V}), f_{m_s}^s(\vec{V}) \in F, \hat{g}_{jl}^S \in [g_{jl}^-; g_{jl}^+], j = 1, \dots, m, l = 1, \dots, S$$

де m_s – кількість структурних елементів s-ої інтервальної моделі, $[I_{min}, I_{max}]$ – мінімальне і максимальне значення кількості структурних елементів в моделі, S – кількість моделей-претендентів, F – множина потенційних структурних елементів моделі, $\hat{g}_{jl}^S$ – вектор оцінок параметрів моделі-претендента з структурою λ_s.

Ітераційна процедура для методів оцінювання розв'язків ІСНАР для окремої s-ої моделі претендента ґрунтується на кожній ітерації якості оцінки параметрів математичної моделі. Якість оцінки параметрів, задаємо величиною $\delta(\hat{\hat{g}}_l)$ у вигляді різниці центрів найбільш віддалених між собою модельованого та експериментального інтервалів для кожної часової дискрети, якщо ці інтервали не перетинаються. У випадку перетину цих інтервалів, функцію $\delta(\hat{\hat{g}}_l)$ визначатимемо найменшою шириною їх перетину. Вираз для функції $\delta(\hat{\hat{g}}_l)$ представимо у такому вигляді:

$$\delta(\hat{\hat{g}}_l) = \max_{i=1,\dots,N} \{|mid([\hat{v}_k]) - mid([z_k^-; z_k^+])|\},$$
якщо $[\hat{v}_k] \cap [z_k^-; z_k^+] = \emptyset, \exists k = 0, \dots, K \quad (8)$

$$\delta(\hat{\hat{g}}_l) = \max_{i=1,\dots,N} \{wid([\hat{v}_k]) - wid([\hat{v}_k] \cap [z_k^-; z_k^+])\},$$
якщо $[\hat{v}_k] \cap [z_k^-; z_k^+] = \emptyset, \forall k = 0, \dots, K \quad (9)$

Розглянемо основні фази методу структурної ідентифікації моделей динамічних об'єктів на основі поведінкових моделей бджолиної колонії [5 – 8].

Першим кроком алгоритму є введення початкових змінних: інтервальні дані отримані в ході проведення експерименту; параметри алгоритму структурної ідентифікації, такі як S – кількість моделей-претендентів, MCN – загальна кількість ітерацій, LIMIT – число, яке визначає вичерпність джерела, $[I_{min}, I_{max}]$ – мінімальне і максимальне значення кількості структурних елементів в моделі, F – множину потенційних структурних елементів моделі; параметри параметричної ідентифікації, такі як S – чисельність усієї популяції бджіл, MCN – загальна кількість ітерацій, LIMIT – число, яке визначає вичерпність джерела, умови зупинки алгоритму, послідовну чи паралельну стратегію виконання, та нижню і верхню межу можливих значень вектор оцінок параметрів моделі-претендента.

На фазі ініціалізації формуємо початкову множину моделей-претендентів Λ_0 потужністю S випадковим чином із набору структурних елементів F. За допомогою алгоритму параметричної ідентифікації для кожної моделі обчислюємо значення функції мети.

Фаза робочих бджіл відповідає за синтез множини поточних моделей Λ'_{mcn}. На основі моделі λ_s формуємо нову модель λ'_s, яка є "околом". При цьому нову модель λ'_s формуємо у спосіб випадкового вибору і заміни частини структурних елементів поточної моделі λ_s. Під час обчислення елементів n_s, які потрібно замінити, враховуємо якість $\delta(\lambda_s)$ поточної моделі і кількість її елементів $m_s \in [I_{min}, I_{max}]$. На цій же фазі проводимо селекцію, для вибору кращої моделі з поточної і генерованої.

В контексті задачі структурної ідентифікації фази бджіл дослідників означає визначення околу дослідження поточної моделі λ_s. Для генерації околу використовуємо описаний вище метод. Для визначення кількості моделей в околі будемо використовувати ймовірнісний підхід, який описаний формулою (10):

$$P_s(\lambda_s^1) = \frac{1 - \delta(\lambda_s^1)}{\sum_{s=1}^{S}(1 - \delta(\lambda_s^1))}, s = 1..S \quad (10)$$

Отримуємо точні значення кількості новостворених моделей в околі $m_s = int(P_s * S)$. Для кожної моделі введемо показник limit, що відповідає за вичерпність джерела нектару в контексті поведінкової моделі бджолиної колоній, і слугує умовою виходу із локальних мінімумів в контексті розв'язання оптимізаційної задачі структурної ідентифікації. Далі на цій фазі проводимо групову селекцію для кожного сформованого околу поточної моделі λ_s. Значення лічильника limit збільшуємо на одиницю кожного разу якщо при груповій селекції поточна модель не оновилася.

Наступним кроком обираємо модель з найменшим значенням функції мети $\delta(\lambda_s)$. Якщо $\delta(\lambda_s) = 0$, завершуємо процедуру структурної ідентифікації, в протилежному випадку переходимо до наступної фази.

Фаза бджіл розвідників реалізовує механізм виходу з локальних мінімумів. Якщо значення лічильника limit поточної моделі λ_s перевищило значення змінної LIMIT, вказаної на етапі ініціалізації, тоді ця модель вважається локальним мінімум і її потрібно замінити. Нову модель генеруємо випадковим чином, так само як на фазі ініціалізації.

Архітектура програмного забезпечення

Архітектура програмного забезпечення організована в три основні шари, відповідно до сучасних принципів архітектури програмного забезпечення та патернам проєктування [11 – 13]:

– Графічний інтерфейс користувача (UI): Надає інтерактивне середовище для користувачів для конфігурації параметрів ідентифікації моделей, управління проєктами, імпорту/експорту даних та візуалізації результатів. UI розроблений для зручності використання та доступності, абстрагуючи складність базових алгоритмів.

– Шар бізнес-логіки (C# бібліотека): Інкапсулює реалізацію алгоритму ШБК, підтримуючи як параметричну, так і структурну оптимізацію [2 – 8, 14 – 15]. Цей шар управляє робочим процесом оптимізації (7), перевіркою даних та організовує паралельні обчислення на CPU або GPU.

– Nvidia CUDA бекенд: Відповідає за інтенсивні обчислювальні завдання, переносячи обчислення функції якості оцінки параметрів математичної моделі (8, 9) на графічний процесор [8, 16 – 19], а саме динамічно генерує, компілює та виконує CUDA-ядра, адаптовані до специфічної проблеми оптимізації, значно прискорюючи процес ідентифікації для великомасштабних проблем.

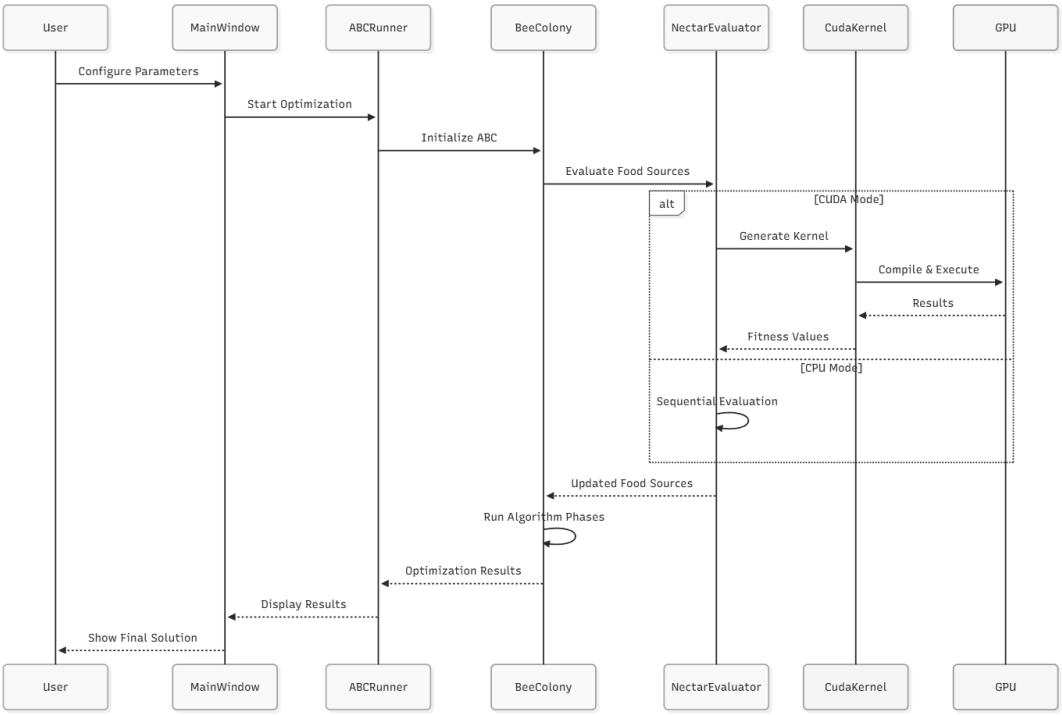

Рис. 1. Діаграма послідовності взаємодії компонентів

Система слідує чіткому та логічному потоку даних, реалізуючи сучасні патерни паралельних обчислень [10, 20], що відображено на діаграмі послідовності на рис. 1:

1. Введення користувача: користувач конфігурує проблему оптимізації через користувацький інтерфейс, рис. 2 та рис. 3.
2. Перевірка параметрів: основна бібліотека перевіряє на валідність та попередньо обробляє вхідні дані.
3. Виконання оптимізації (7): бібліотека ініціює алгоритм ШБК [8], вибираючи відповідну стратегію виконання (послідовну, паралельну CPU або CUDA).
4. Обчислення: якщо CUDA доступна та вибрана, бекенд генерує та компілює специфічне для проблеми ядро, потім виконує його на GPU.
5. Обробка результатів: результати збирають, пост-обробляють та представляють користувачу через UI, включаючи візуалізації та опції експорту, рис. 4.

Графічний інтерфейс користувача реалізований з використанням Windows Forms, надаючи знайоме та відгукливе середовище для користувачів. Користувачі можуть зберігати та завантажувати проєкти оптимізації, забезпечуючи відтворюваність та сприяючи співпраці. Підтримує формати CSV та JSON для вхідних та вихідних даних, що дозволяє інтеграцію з зовнішніми інструментами та наборами даних. Панелі накладання та індикатори стану надають зворотний зв'язок під час довгострокових обчислень.

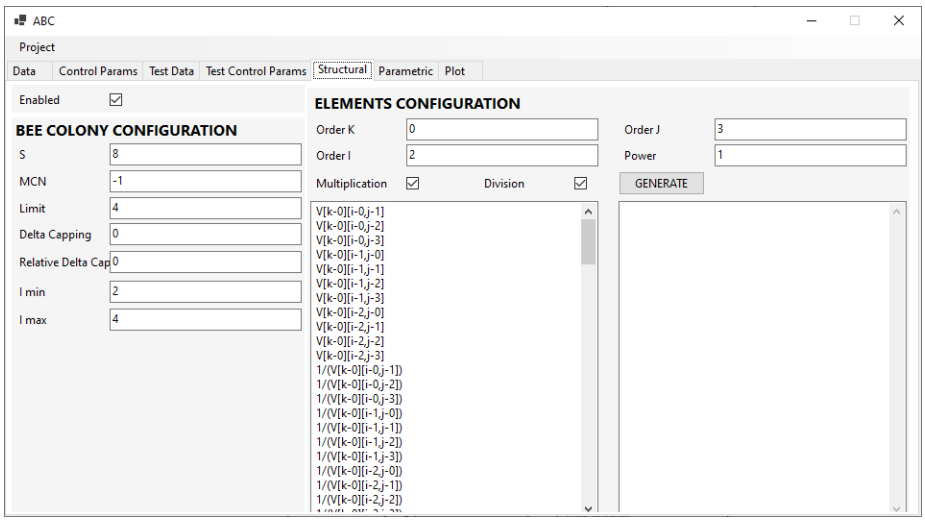

Рис. 2. Введення параметрів для структурної ідентифікації

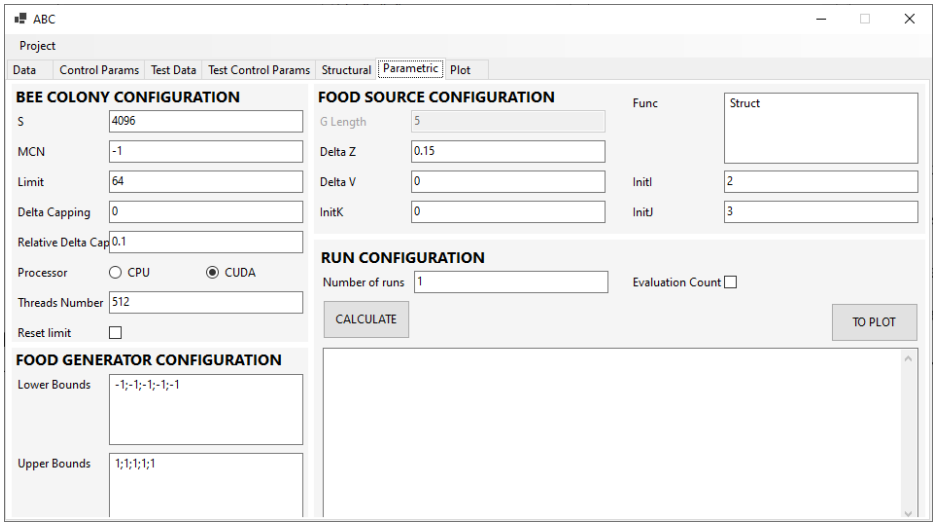

Рис. 3. Введення параметрів для параметричної ідентифікації

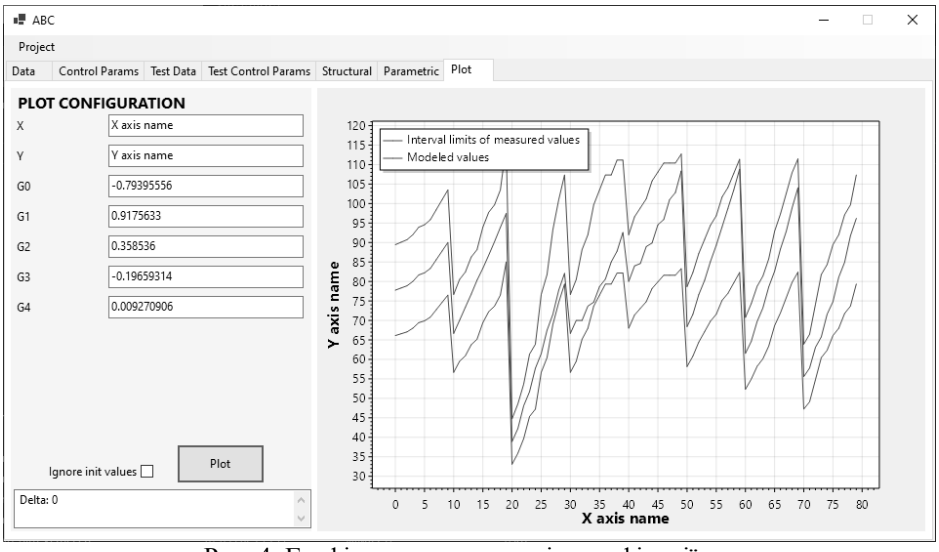

Рис. 4. Графік з результатами ідентифікації

Користувацький інтерфейс відокремлений від основної обчислювальної логіки і відповідає лише за налаштування параметрів і обробку вхідних даних. Це розділення покращує підтримуваність та робить можливими майбутні покращення UI або повністю його заміну на такі як веб-базовані або крос-платформені інтерфейси [21 – 22]. Відносини між компонентами UI та іншими шарами ілюструються в діаграмі ієрархії класів на рис. 5.

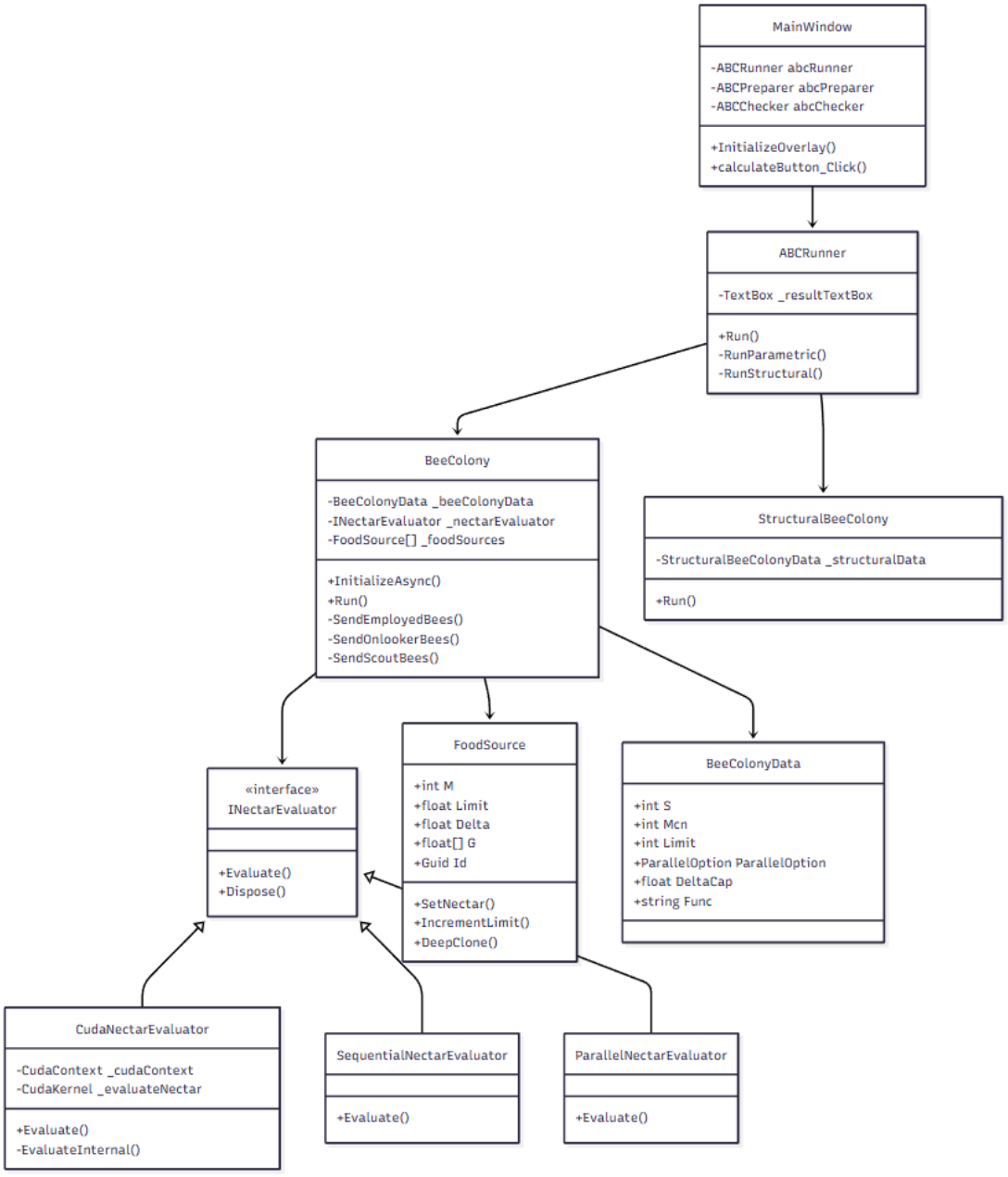

Рис. 5. Діаграма ієрархії класів

Основна бібліотека відповідає за реалізацію алгоритму ШБК та управління робочим процесом оптимізації. Її основні відповідальності включають:

– Реалізація алгоритму: Підтримує як параметричну оптимізацію (налаштування параметрів моделі), так і структурну оптимізацію (модифікація структури моделі) [2 – 8, 23 – 24]. Алгоритм організований у різні фази: ініціалізація, фаза працюючих бджіл, фаза бджіл-спостерігачів, фаза бджіл-розвідників та перевірка збіжності [2 – 8, 25, 26].

– Управління даними: Система використовує спеціалізовані структури даних для представлення проблем оптимізації, кандидатів рішень, параметрів алгоритму та результатів,

відповідно до сучасних патернів проєктування програмного забезпечення [9, 11]. Забезпечує цілісність та консистентність даних протягом процесу оптимізації.

– Вибір стратегії виконання: Користувач може вибрати найбільш відповідну стратегію виконання на основі розміру проблеми та доступного апаратного забезпечення (послідовну, паралельну CPU або CUDA GPU) [8, 27].

– Розширюваність: Модульний дизайн з чіткими інтерфейсами дозволяє інтеграцію нових стратегій оптимізації, методів оцінки або форматів даних з мінімальними змінами наявного коду [21, 22].

Шар бізнес-логіки розроблений для гнучкості та розширюваності, відповідно до встановлених патернів проєктування програмного забезпечення [11]. Інкапсулюючи алгоритмічну складність та експонуючи чіткі інтерфейси, система може бути легко адаптована до нових проблемних областей або розширена додатковими техніками оптимізації [9, 11]. Ієрархія класів та відносини детально описані на рис. 5.

Nvidia CUDA надає високопродуктивні паралельні обчислення для оцінки функції мети [2 – 8], що є найбільш обчислювально інтенсивною частиною процесу ідентифікації моделей [10, 28]. Переносячи обчислення на GPU, система досягає значних прискорень для великомасштабних проблем [8, 17, 18]. Основні переваги розробленого забезпечення:

– Динамічна генерація ядер: Система динамічно генерує CUDA ядра на основі параметрів проблеми та користувацько-визначених функцій, відповідно до сучасних практик GPU-обчислень [7, 19, 20, 29, 30]. Процес генерації ядер включає завантаження шаблонів, компіляцію з використанням NVIDIA Runtime Compiler (NVRTC) та генерацію PTX байткоду для виконання на GPU. Використання динамічної генерації ядер забезпечує адаптацію системи до різноманітних типів проблем без необхідності ручної розробки ядер для кожного випадку, рис. 6.

– Ефективне управління пам'яттю: Управляє розподілом GPU-пам'яті, передачею даних та очищенням ресурсів для максимізації продуктивності та запобігання витоків пам'яті [16, 17].

– Підтримка інтервальної арифметики: Дозволяє надійну оптимізацію в присутності невизначеності або інтервальних даних [13].

– Підтримка багатовимірності: Обробляє одно-, дво- і тривимірні проблеми, що робить систему застосовною до широкого діапазону завдань ідентифікації.

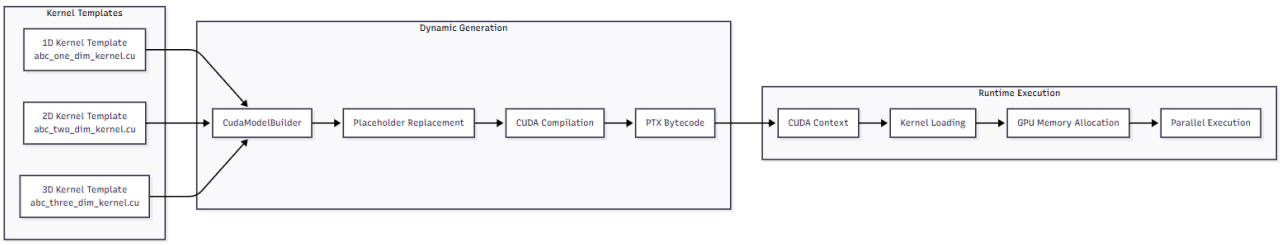

Рис. 6. Динамічна генерація CUDA ядер

Користувачі можуть імпортувати дані у форматах CSV або JSON, які потім проходять валідацію та трансформуються у внутрішні представлення, придатні для оптимізації, а також використовувати введення UI для ручного заповнення. Система підтримує збереження та завантаження повних станів проєктів, включаючи параметри, дані та результати, у форматі JSON. Це сприяє відтворюваності та співпраці. Результати ідентифікації відображаються за допомогою графіків та таблиць, що дозволяє користувачам ефективно інтерпретувати та аналізувати результати оптимізації.

Розроблене програмне забезпечення має комплексну перевірку вхідних даних, щоб гарантувати, що на оптимізацію надходять лише правильно сформовані задачі. Це зменшує ризик помилок під час виконання. Система включає журналювання та моніторинг продуктивності, відповідно до найкращих практик наукових обчислень [22, 32]. Система

журналювання надає можливості замірів часу виконання для критичних операцій, конфігурований вивід та збір метрик продуктивності протягом процесу ідентифікації. Надійна обробка винятків забезпечує відновлення від помилок, особливо в контексті GPU обчислень, де управління ресурсами є критичним.

Висновки

Представлена архітектура демонструє надійний, масштабований та розширюваний підхід до реалізації передових алгоритмів оптимізації. Ключовим досягненням є унікальна інтеграція структурної та параметричної оптимізації в єдиному фреймворку, що дозволяє вирішувати проблеми ідентифікації складних моделей. Інноваційний підхід до динамічної генерації CUDA ядер забезпечує оптимальну продуктивність для різноманітних типів проблем без попередньої розробки спеціалізованих ядер. Система надає потужну платформу для дослідників через забезпечення відтворюваності експериментів та підтримку різноманітних математичних моделей. Для інженерних застосувань забезпечує високопродуктивну обробку великомасштабних проблем з інтуїтивним інтерфейсом та надійною обробкою помилок. Технічні інновації включають тришарову архітектуру, динамічну компіляцію функцій та адаптивне управління GPU-пам'яттю. Поточні обмеження включають залежність від NVIDIA GPU та обмежену підтримку розподілених обчислень. Майбутній розвиток спрямований на підтримку альтернативних GPU-архітектур та реалізацію хмарних обчислень.

СПИСОК ЛІТЕРАТУРИ

1. Karaboga D. An idea based on honey bee swarm for numerical optimization : Technical report. Erciyes University, Engineering Faculty, Computer Engineering Department. Erciyes University, 2005. 10 p. URL: https://abc.erciyes.edu.tr/pub/tr06_2005.pdf.

2. Artificial Bee Colony Algorithm with Modified Operators of Determining the Profitable Food Sources for Identification the Models of Atmospheric Pollution by Nitrogen Dioxide / M. Dyvak et al. *Proceedings of the 2020 10th International Conference on Advanced Computer Information Technologies (ACIT)*. 2020. P. 122–125. DOI: 10.1109/ACIT49673.2020.9208901.

3. Modeling Based on the Analysis of Interval Data of Atmospheric Air Pollution Processes with Nitrogen Dioxide due to the Spread of Vehicle Exhaust Gases / M. Dyvak et al. *Sustainability*. 2023. Vol. 15. P. 2163.

4. Convergence Estimation of a Structure Identification Method for Discrete Interval Models of Atmospheric Pollution by Nitrogen Dioxide / I. Darmorost et al. *Proceedings of the 2019 9th International Conference on Advanced Computer Information Technologies (ACIT)*. Ceske Budejovice, Czech Republic, 2019. P. 117–120.

5. Dyvak M. Parameters Identification Method of Interval Discrete Dynamic Models of Air Pollution Based on Artificial Bee Colony Algorithm. *Proceedings of the 2020 10th International Conference on Advanced Computer Information Technologies (ACIT).* Deggendorf, Germany, 2020. P. 130–135.

6. Features of structure identification the macromodels for nonstationary fields of air pollutions from vehicles / N. Ocheretnyuk et al. *Proceedings of the 11th International Conference on Modern Problems of Radio Engineering, Telecommunications and Computer Science*. 17–19 May 2012. P. 444.

7. Parallel Computations in the Problem of Identification of Interval Discrete Models based on Swarm Intelligence of a Bee Colony / M. Dyvak et al. *Proceedings of the 2023 13th International Conference on Advanced Computer Information Technologies (ACIT)*. 2023. P. 23–28.

8. Dyvak M., Kindzerskyi O. Implementation of Parallel Computation for Identification of Interval Models based on Multi-core Parallelism and CUDA Technology. *Proceedings of the 2024 IEEE International Conference on Advanced Computer and Information Technologies (ACIT)*. 2024. P. 72–76. DOI: 10.1109/ACIT62333.2024.10712545.

9. Bass L., Clements P., Kazman R. Software Architecture in Practice. 3rd ed. Boston : Addison-Wesley, 2012. 624 p.

10. Rodriguez M., Garcia J. Parallel metaheuristics: Current trends and future directions. *Swarm and Evolutionary Computation*. 2022. Vol. 68. P. 100–115.

11. Design Patterns: Elements of Reusable Object-Oriented Software / Gamma E., Helm R., Johnson R., Vlissides J. Boston : Addison-Wesley, 1994. 395 p.

12. Thompson E., Brown K. Software architecture for high-performance computing: Modern approaches and best practices. *IEEE Transactions on Software Engineering*. 2024. Vol. 50, № 1. P. 78–95.

13. Anderson M., Wilson P. Interval arithmetic in optimization: Theory and applications. *Applied Mathematics and Computation*. 2023. Vol. 456. P. 1–18.

14. Kumar A., Kumar D. A comprehensive review of artificial bee colony algorithm variants. *Swarm and Evolutionary Computation*. 2019. Vol. 44. P. 1–15.

15. Li X., Yang G. GPU-accelerated artificial bee colony algorithm for large-scale optimization problems. *Journal of Parallel and Distributed Computing*. 2020. Vol. 142. P. 1–12.

16. Sanders J., Kandrot E. CUDA by Example: An Introduction to General-Purpose GPU Programming. Boston : Addison-Wesley, 2010. 296 p.

17. Scalable parallel programming with CUDA / J. Nickolls et al. 2008. Vol. 6, № 2. P. 40–53.

18. Zhang Y., Liu X. CUDA-based optimization frameworks: A systematic review. *Journal of Supercomputing*. 2023. Vol. 79, № 4. P. 1234–1256.

19. Patel A., Singh R. Dynamic compilation techniques for GPU computing: Recent advances and applications. *ACM Computing Surveys*. 2023. Vol. 56, № 2. P. 1–28.

20. Johnson R., Smith T. Extensible optimization frameworks: Design patterns and implementation strategies. *Software: Practice and Experience*. 2024. Vol. 54, № 3. P. 234–251.

21. Williams D., Davis M. Performance monitoring and logging in scientific computing applications. *Journal of Computational Science*. 2023. Vol. 67. P. 1–12.

22. Lee S., Kim J. Structural optimization using swarm intelligence: A comprehensive survey. *Engineering Applications of Artificial Intelligence*. 2024. Vol. 127. P. 1–22.

23. Martinez C., Lopez A. Real-time optimization systems: Architecture and implementation challenges. *Journal of Systems and Software.* 2023. Vol. 195. P. 1–15.

24. Cantú-Paz E. Efficient and Accurate Parallel Genetic Algorithms. Berlin : Springer, 2000. 273 p.

25. Grefenstette J. J. Optimization of control parameters for genetic algorithms. *IEEE Transactions on Systems, Man, and Cybernetics*. 1986. Vol. 16, № 1. P. 122–128.

26. Дивак М., Кіндзерський О. Дослідження ефективності паралельної обчислювальної схеми ідентифікації інтервальних дискретних моделей на основі ройового інтелекту. *Herald of Khmelnytskyi National University. Technical Sciences*. 2024. Vol. 331, № 1. P. 29–37.

27. Kennedy J., Eberhart R. Particle swarm optimization. *Proceedings of IEEE International Conference on Neural Networks*. 1995. Vol. 4. P. 1942–1948.

28. Goldberg D. E. Genetic Algorithms in Search, Optimization and Machine Learning. Boston : Addison-Wesley, 1989. 412 p.

29. Dorigo M., Birattari M., Stutzle T. Ant colony optimization. *IEEE Computational Intelligence Magazine*. – 2006. Vol. 1, № 4. P. 28–39.

30. Chen H., Wang L. Modern software architecture patterns for scientific computing applications. *IEEE Software*. 2021. Vol. 38, № 3. P. 45–52.

31. Talbi E. G. Metaheuristics: From Design to Implementation. Hoboken : John Wiley & Sons, 2009. 500 p.

32. Alba E., Tomassini M. Parallelism and evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*. 2002. Vol. 6, № 5. P. 443–462.

33. Taylor B., Anderson C. Modern C# development for scientific applications: Best practices and patterns. *ACM Computing Surveys*. 2024. Vol. 57, № 1. P. 1–25.

34. Garcia L., Rodriguez P. Cloud-native optimization systems: Architecture and deployment strategies. *IEEE Cloud Computing*. 2023. Vol. 10, № 2. P. 45–62.

Стаття надійшла до редакції 22.06.2025.
Стаття пройшла рецензування 28.06.2025.

Дивак Микола Петрович – д-р техн. наук, професор, проректор з наукової роботи.

Кіндзерський Олександр Віталійович – аспірант, e-mail: o.kindzerskyy@gmail.com.
Західноукраїнський національний університет.