УДК 681.5.

П.А. Іваненко, А.Ю. Дорошенко, О.М. Овдій, Л.М. Суслова

АВТОТЬЮНЕР ТА ВІЗУАЛІЗАЦІЯ ДЛЯ ЗАДАЧІ МЕТЕОРОЛОГІЧНОГО ПРОГНОЗУВАННЯ

Розглядається застосування техніки автотьюнінгу та розширення картографічного інструментарію Google Maps для комплексного розв'язання задачі короткотермінового моделювання циркуляції атмосфери. Наводяться чисельні показники швидкодії розробленого паралельного алгоритму.

Вступ

Задача метеорологічного прогнозування є типовим прикладом складної прикладної обчислювальної задачі з високими вимогами до точності отриманих результатів та жорсткими часовими обмеженнями. Підвищення очікуваної межі точності розв'язку значно збільшує обсяг виконуваних обчислень, що неухильно примножує часові витрати на розв'язання задачі тому оптимальний алгоритм вимагає відшукання компромісу.

Дана робота розглядає паралельну реалізацію моделі короткотермінового прогнозу метеорологічних умов, процес її оптимізації з використанням методології автотьюнінгу [1], а також візуалізацію результатів обчислень за допомогою картографічного інструментарію Google Maps з використанням оригінального алгоритму побудови ізоліній. Робота продовжує дослідження, представлені у [2–4] й, зокрема, демонструє ефективність застосування розглянутих підходів для комплексного вирішення складної наукової задачі.

Мета роботи – створення універсального й оптимального з точки зору швидкодії та точності паралельного алгоритму. Під універсальністю розуміється вимога отримання максимально високих результатів швидкодії у будь-якому обчислювальному середовищі. Досягнення цієї мети значно спрощує використання методології автотьюнінгу.

Структура роботи є наступною: перший розділ містить постановку задачі й опис математичної моделі; у розділі 2 розглянуто процес створення паралельного алгоритму що розв'язує поставлену задачу; третій розділ присвячено оптимізації побудованого алгоритму за допомогою автоматизованого інструментарію [3]. У розділах 4 і 5 представлені результати проведеної оптимізації та візуалізація результатів прогнозування відповідно.

1. Постановка задачі

В роботі розглядається задача динамічної метеорології, а саме задача моделювання циркуляції атмосфери. Метою є побудова максимально ефективного алгоритму реалізації моделі короткочасного (від години до декількох днів) прогнозу стану атмосфери а також візуалізація результатів обчислень.

Процеси, що описує модель, класифікуються як макромасштабні − їхній характерний горизонтальний розмір знаходиться в межах від кількох сотень до кількох тисяч кілометрів. Як приклад макромасштабних рухів можна навести синоптичні вихорі (циклони та антициклони, улоговини та гребні), поля хмар, довгі хвилі (хвилі Росбі [5]), пасатні та мусонні течії, західне та східне перенесення тощо.

Розглянемо коротко основні характеристики математичної моделі. Детальний опис можна знайти у [6].

Рельєф суттєво впливає на процеси та явища, що мають місце в нижній частині атмосфери. Розглянута модель враховує вплив рельєфу шляхом переходу до нової вертикальної координати, яка неявно пов'язана із орографією земної поверхні. Неявний зв'язок має місце в наслідок вибору вертикальною координатою нормованого тиску: $\sigma = p/p_0$, де p_0 – приземний тиск.

Модель використовує сферичну систему координат (рис. 1). Введемо змінні що представляють географічні координати точки: довгота λ, широта φ та висота над рівнем моря $z = r - a$, де r – відстань від центру Землі до поточної точки, $a = 6.373 \cdot 10^6$ *м* – середній радіус Землі.

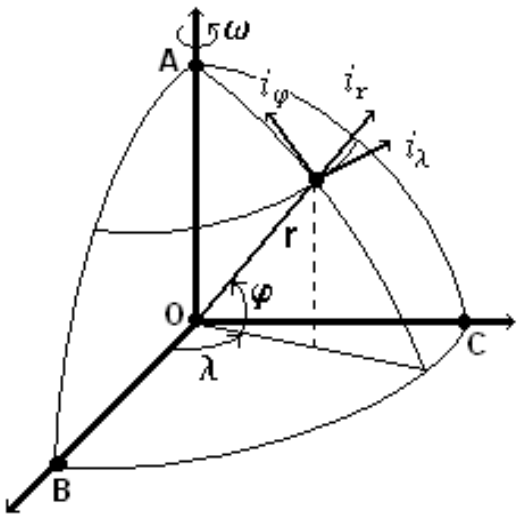

Рис. 1. Неінерційна система відліку, що пов'язана із географічними координатами

Розглядатимемо вектор швидкості $V = (u, v, w)$, де u, v, w є проекціями вектора V у локальній прямокутній системі координат. Вісі цієї системи такі: i_λ – направлена по дотичній до широтного кола із заходу на схід, вісь i_φ – по дотичній до меридіана від екватора до полюса, i_r – по місцевій вертикалі догори. Невідомими величинами в нашій моделі є: вектор швидкості V, абсолютна температура середовища T та густина ρ повітря. Значення тиску p вважатимемо відомими, як і початково-крайові умови впродовж усього часу розв'язання задачі.

Модель складається із трьох еволюційних рівнянь конвективної дифузії (1), (2) та (3) для визначення горизонтальних складових вітру та температури відповідно формул:

$$\frac{\partial u}{\partial t} + \frac{u}{a\cos\varphi}\frac{\partial u}{\partial \lambda} + \frac{v}{a}\frac{\partial u}{\partial \varphi} + w\frac{\partial u}{\partial z} =$$

$$= -\frac{1}{\rho a\cos\varphi}\frac{\partial p}{\partial \lambda} + \frac{1}{\rho}\frac{\partial}{\partial \lambda}\left(\frac{\rho\mu_{\lambda\varphi}}{(a\cos\varphi)^2}\frac{\partial u}{\partial \lambda}\right) +$$

$$+ \frac{1}{\rho}\frac{\partial}{\partial \varphi}\left(\frac{\rho\mu_{\lambda\varphi}}{a^2}\frac{\partial u}{\partial \varphi}\right) + \frac{1}{\rho}\frac{\partial}{\partial z}\left(\rho\mu_z\frac{\partial u}{\partial z}\right) +$$

$$+ (v\sin\varphi - w\cos\varphi)\left(2\omega + \frac{u}{a\cos\varphi}\right), \quad (1)$$

$$\frac{\partial v}{\partial t} + \frac{u}{a\cos\varphi}\frac{\partial v}{\partial \lambda} + \frac{v}{a}\frac{\partial v}{\partial \varphi} + w\frac{\partial v}{\partial z} =$$

$$-\frac{1}{\rho a}\frac{\partial p}{\partial \varphi} + \frac{1}{\rho}\frac{\partial}{\partial \lambda}\left(\frac{\rho\mu_{\lambda\varphi}}{(a\cos\varphi)^2}\frac{\partial v}{\partial \lambda}\right) +$$

$$+ \frac{1}{\rho}\frac{\partial}{\partial \varphi}\left(\frac{\rho\mu_{\lambda\varphi}}{a^2}\frac{\partial v}{\partial \varphi}\right) + \frac{1}{\rho}\frac{\partial}{\partial z}\left(\rho\mu_z\frac{\partial v}{\partial z}\right) -$$

$$- u\left(2\omega + \frac{u}{a\cos\varphi}\right)\sin\varphi - \frac{wv}{a}, \quad (2)$$

де $\mu_{\lambda\varphi}$ та μ_z – коефіцієнти горизонтального та вертикального турбулентних обмінів.

$$\frac{dT}{dt} = \frac{1}{\rho}\frac{\partial}{\partial \lambda}\left(\frac{\rho\mu_{\lambda\varphi}}{(a\cos\varphi)^2}\frac{\partial T}{\partial \lambda}\right) +$$

$$+ \frac{1}{\rho}\frac{\partial}{\partial \varphi}\left(\frac{\rho\mu_{\lambda\varphi}}{a^2}\frac{\partial T}{\partial \varphi}\right) + \frac{1}{\rho}\frac{\partial}{\partial z}\left(\rho\mu_z\frac{\partial T}{\partial z}\right). \quad (3)$$

Крайові умови забезпечуються інтерполяцією за часом вхідних метеорологічних даних, що вже перетворені до потрібного просторового формату. Початковими умовами є або результати попереднього часового кроку, або початкові умови задачі.

Обчислення за діагностичними рівняннями для знаходження вертикальної компоненти вектора V для визначення вертикальної складової вітру та густини повітря (рівняння Клапейрона) проводяться після визначення решти невідомих метеовеличин.

Поле тиску у всій просторово-часовій області визначення задачі вважається відомим. Для цього здійснюється часова інтерполяція метеоданих тиску.

Для постановки початкових та крайових умов моделі використовувався електронний архів даних об'єктивного аналізу полів метеовеличин, що отриманий із регіонального центру прогнозування погоди „Офенбах". Формат цих метеорологічних даних є наступним:

- вертикальна координата – тиск;
- вертикальна протяжність від 1000 до 50 гПа;
- просторова область 0° – 90° пвн. ш. та 0° – 90° сх. д.;
- просторова дискретність 1.5° (за широтою та довготою);
- часова дискретність 12 годин.

2. Паралельний алгоритм

В межах цієї роботи розглядається спрощена двовимірна задача з обрахуванням тільки швидкості та напрямку вітру. Практично спрощена програма дозволяє отримати прогноз для одного горизонтального рівня повної тривимірної моделі.

Модель розпаралелюється на трьох рівнях – за еволюційними рівняннями, на рівні даних (геометричний паралелізм) та за допомогою модифікованого адитивно-усередненого методу розщеплення (далі МАУМ) [6–8].

Паралелізм за рівняннями. Розглянута модель дає можливість розв'язувати рівняння (1) та (2) незалежно. За рахунок цього так званого «високорівневого паралелізму» обмін результатами розв'язків рівнянь виконується через вхідні данні на кожній ітерації (алгоритм є ітеративним).

Метод розщеплення. Як вже було сказано раніше – загальна схема виконуваних обрахунків є ітеративною. Вхідні данні задають стан атмосфери на момент часу t_0. Для отримання прогнозу на деякий момент $T > t_0$ часовий відрізок розбивається на n порівняно коротких відрізків протяжністю Δt. Кожної ітерація обчислює стан атмосфери через час Δt і її вихідні дані стають вхідними (початковими) даними для наступної ітерації (рис. 2).

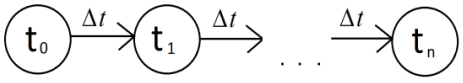

Рис. 2. Ітеративна схема обчислень

Збільшення кроку Δt призводить до зменшення кількості ітерацій а тому й загального обсягу обчислень, проте призводить до збільшення похибки прогнозу тому рекомендується при обрахунках обирати часовий крок у декілька десятків секунд.

В межах кожної ітерації незалежно розв'язуються рівняння за кожним з координатних напрямків після чого виконується збір і обробка результатів, що вимагає синхронізації. Використання МАУМ дозволяє поєднати декілька ітерації (позначимо їх кількість m) в одну. В результаті збір та усереднення результатів виконується тільки в кінці такої «склейки», що зменшує часові затрати на синхронізацію обчислень.

Слід зауважити, що збільшення значення параметра m також призводить до втрати точності розв'язку й рекомендований діапазон значень є $m \in [1..10]$ [6].

Оскільки для розв'язання рівнянь (1) – (3) використовується однаковий чисельний метод, то часові витрати у кожній гілці на протязі всього m-циклу є однакові, що дає можливість оптимально розподілити обчислення для паралельних обчислювачів з гомогенною архітектурою.

Геометричний паралелізм. Окрім незалежного розв'язання кожного з рівнянь моделі є можливість геометричної декомпозиції області обчислень за кожним координатним напрямком.

Позначимо S кількість підобластей. Тоді для проекції вектора швидкості u (1) у двовимірному варіанті задачі схема обчислення для одного m-циклу буде виглядати наступним чином (рис. 3):

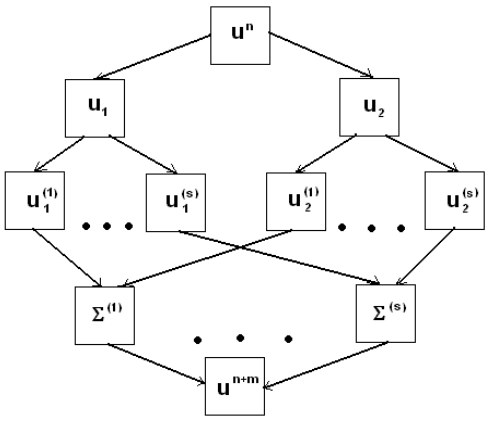

Рис. 3. Схема розв'язання одного еволюційного рівняння протягом m-циклу

Як видно з діаграми, для кожного рівняння моделі декомпозиція області обчислень дозволяє створити $2 \times S$ незалежних між собою підзадач.

Програмна реалізація алгоритму метеопрогнозування виконана засобами Java. Вона розрахована на виконання на обчислювачі з моделлю спільної пам'яті й використовує засоби пакету Java concurrent для організації паралельних обчислень. Кожна підзадача виконується в окремому потоці. Кількість використовуваних потоків є фіксованою.

3. Оптимізація розробленого алгоритму

В попередньому розділі описана трирівнева схема виконуваних паралельних обчислень. Розглянемо параметри, що впливають на загальну ефективність паралельної програми.

Більшість паралельних алгоритмів, орієнтованих на геометричний паралелізм мають наступну схему розподілу часу обчислень (рис. 4):

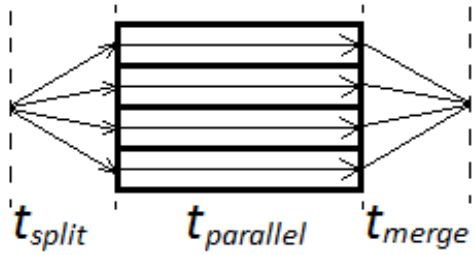

Рис. 4. Типовий розподіл часових витрат для геометричного паралелізму

t_{split} та t_{merge} – час, затрачений на розподіл вхідних даних та поєднання результатів відповідно. Протягом цих часових відрізків зазвичай виконується синхронізована секція, під час якої більша частина обчислювальних ресурсів простоює. Під t_{split} розуміється час «чисто паралельних» обчислень. Очевидним є те, що для досягнення максимальної ефективності алгоритму необхідно мінімізувати час $t_{split} + t_{merge} \to 0$.

Рис. 4 описує часові затрати для однієї ітерації обчислень нашого алгоритму. Як вже зазначалось в попередньому розділі, застосування МАУМ дозволяє виконувати розподіл і збір результатів обчислень один раз на m ітерацій, що безумовно підвищує загальну ефективність алгоритму за рахунок незначного збільшення похибки результату (за умови утримання значення параметра m у рекомендованих межах).

Параметр S геометричної декомпозиції впливає на обсяг обчислень в межах кожної підзадачі. Чим менше його значення – тим крупніший «блок даних» й довше паралельна секція, тобто $t_{parallel} >> t_{merge} + t_{split}$. Якщо архітектура обчислювального середовища (далі ОС) є гомогенною, то природнім є прагнення підібрати таке значення S, при якому загальна кількість підобластей буде рівна кількості незалежних обчислювачів (процесорів/ядер) ОС.

Нехай наше ОС здатне обраховувати одночасно N підзадач. Зважаючи на те, що розглянуте спрощення моделі складається з двох еволюційних рівнянь й обраховується прогноз для двовимірного простору, при значенні $S = \frac{N}{4}$ кількість під задач буде достатньою щоб повністю завантажити всі наявні обчислювальні ресурси й розмір підзадач буде максимальним («крупнозерниста» декомпозиція). З іншого боку далеко не завжди $N \vdots 4$. В цьому випадку варто обрати $S > N div 4$.

Як відомо, швидкість доступу до даних процесорного кешу значно швидша за швидкість зчитування їх з оперативної пам'яті. Як вже було вказано в першому розділі – обрана модель характеризується невисоким обсягом вхідних даних. Тому теоретично існує таке значення S, при якому всі дані підзадачі повністю помістяться в кеш процесора й на протязі всього m-циклу взаємодія з порівняно повільною оперативною пам'яттю буде мінімальною. Підбір такого значення S є нетривіальним, оскільки його величина залежить не тільки від обсягу а й від стратегії роботи з кешем. Ці дві характеристики не завжди є відомими й, зважаючи на різноманіття архітектури су-

часних процесорів, сильно різняться для різних ОС. Також їх врахування ускладнює структуру програми, а мануальний підбір емпіричним методом може зайняти багато часу.

Оперування простим параметром S дозволяє значно спростити паралельний алгоритм за рахунок його абстрагування від характеристик архітектури ОС. Для підбору оптимального значення S скористаємося системою TuningGenie [3], яка є програмною реалізацією методології автотьюнінга, який спирається на емпіричну оцінку різних варіантів оптимізовуваної програми. Його загальна схема є наступною (рис. 5).

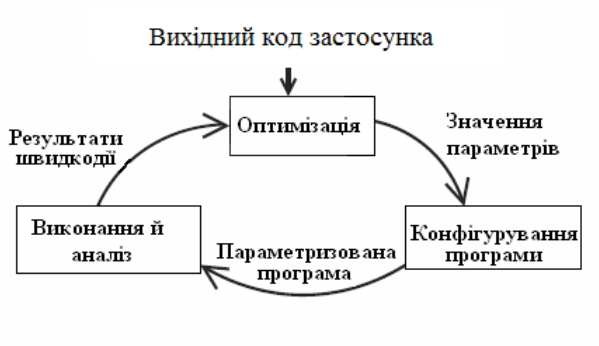

Рис. 5. Типова схема автотьюнінга

Система TuningGenie автоматично генерує автотьюнер для оптимізовуваної програми з використанням метаданих у вихідному коді. Практично достатньо виконати дві модифікації:

- розмістити коментар над внутрішньою змінною програми в якому задаються межі пошуку оптимального значення, наприклад:

```
//tuneAbleParam   name=numSubTasks
//start=1 stop=10 step=2
int numSubTasks = 1;
```

- написати клас-обгортку, який буде запускати головну програму й давати якісну оцінку ефективності виконуваних нею обчислень (в нашому випадку вона залежить від часових витрати).

TuningGenie в автоматизованому режимі емпірично підбере оптимальне значення S в заданих розробником межах (в прикладі numSubTask s $\in$[1,10]). Знайдене значення S – оптимальне для всіх вхідних даних однакової розмірності доки архітектура ОС буде незмінною.

Використання експертних знань розробника суттєво звужує область пошуку оптимальної конфігурації а також звільняє систему автотьюнінга від необхідності аналізу й моделювання обчислень оптимізовуваної програми.

4. Результати експерименту

Дана робота не виконує дослідження адекватності отриманого чисельного розв'язку щодо реальної метеорологічної ситуації. Акцент зроблено на кількісні показники швидкодії програмної реалізації за умови збереження точності розв'язку а також на застосуванні розробленого автоматизованого інструментарію для оптимізації паралельного алгоритму та візуалізації розв'язків. Експеримент виконувався у середовищі з наступними характеристиками (таблиця).

Таблиця. Характеристики експериментального середовища

Операційна система:	Scientific Linux 5
Тип системи	64-бітна ОС
Процесори	2xIntel® Xeon® Processor E5405 Quad Core
Кількість ядер процессора	4
Обсяг L2-кеша	12 Mb
Обсяг оперативної пам'яті	16 Gb

Для проведення експерименту вхідні дані інтерпольовані на більш густу сітку розмірністю 600x600. Часовий крок було обрано рівним 10 секундам. Наступний графік зображує залежність часових затрат на обрахування прогнозу на 24 години від значення параметра S – кількості підобластей геометричної декомпозиції (рис. 6).

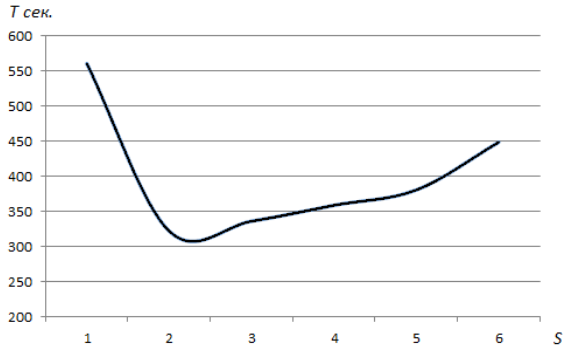

Рис. 6. Результати чисельного експерименту

Значення параметра МАУП покладено $m=10$. Варто зауважити що для пошуку оптимального значення параметра S не виконувалися повне обчислення прогнозу на добу вперед. Для досягнення стабільних результатів замірів достатньо виконати декілька десятків ітерацій m-циклу, що скоротило загальний час автотьюнінгу до декількох годин.

Як видно з рис. 6, оптимальна для тестового ОС величина $S=2$, що є цілком логічним, оскільки при такому значенні обчислення кожної ітерації розбиваються на 8 підзадач, що дорівнює загальній кількості процесорів. За такої конфігурації програма показала мультипроцесорне прискорення $W(8)=3.82$ й ефективність $E(8)=47.75\%$. Раніше висунуте припущення щодо можливого над-прискорення при розбитті на підзадачі невеликої розмірності для тестового ОС експериментально не підтвердилося.

5. Візуалізація результатів обчислень

Для картографічного представлення метеорологічних даних прогнозу за напрямком та силою вітру використано сервіс Google Maps [9], який на даний момент є найпоширенішим геосервісом у світі. Ця потужна платформа надає різноманітну функціональність та зручне API. Зокрема для реалізації використано Google Maps JavaScript API v3 [10] та скриптову мову програмування PHP: Hypertext Preprocessor.

Реалізовано три представлення метеорологічних даних за напрямком та силою вітру:

- за допомогою векторів вітру (рис. 7);
- за допомогою ізоліній сили вітру (ізотах) (рис. 8);
- за допомогою ізообластей сили вітру (рис. 9).

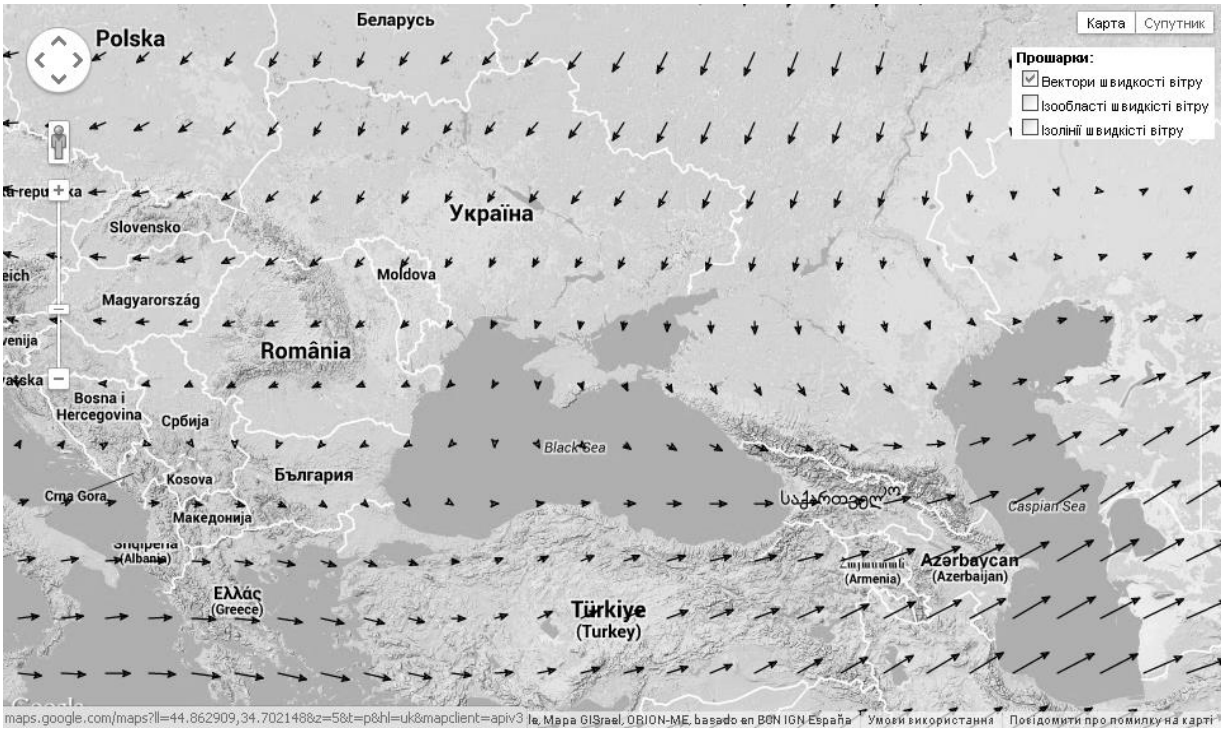

Рис. 7. Картографічне представлення векторів вітру

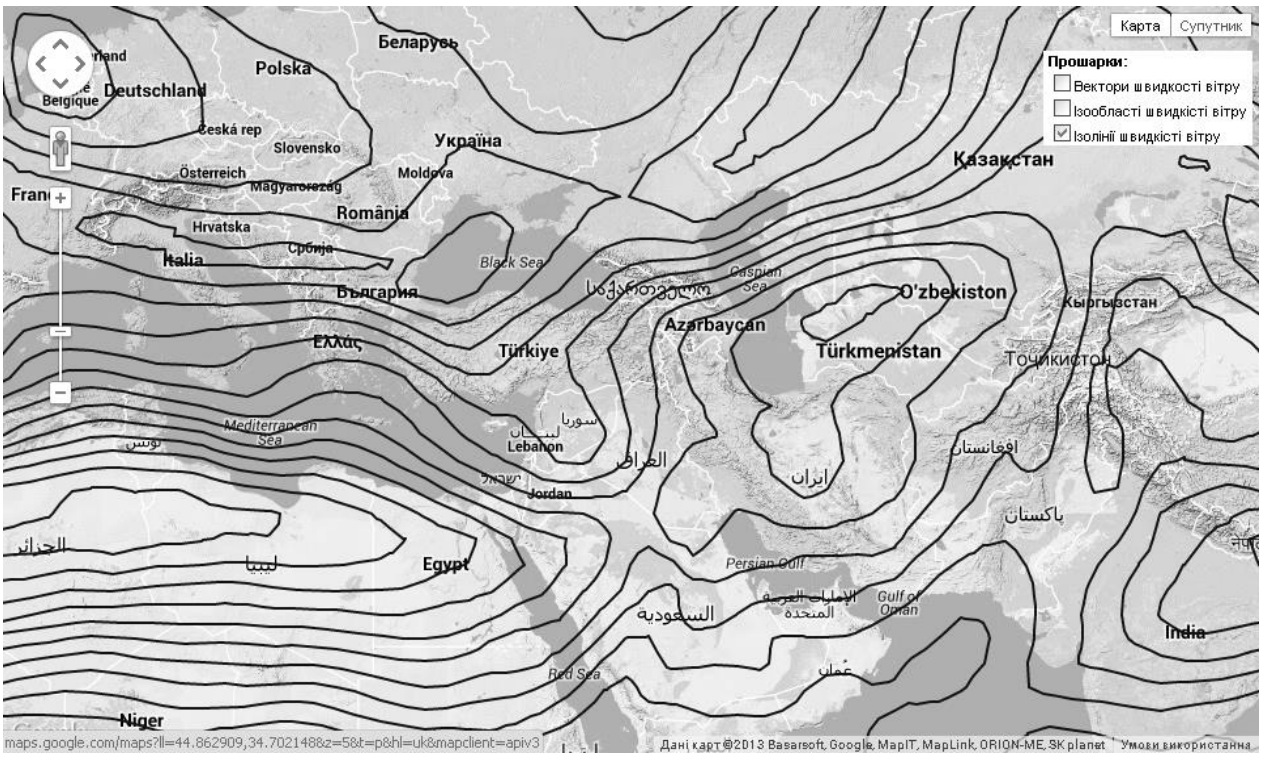

Рис. 8. Картографічне представлення ізоліній сили вітру

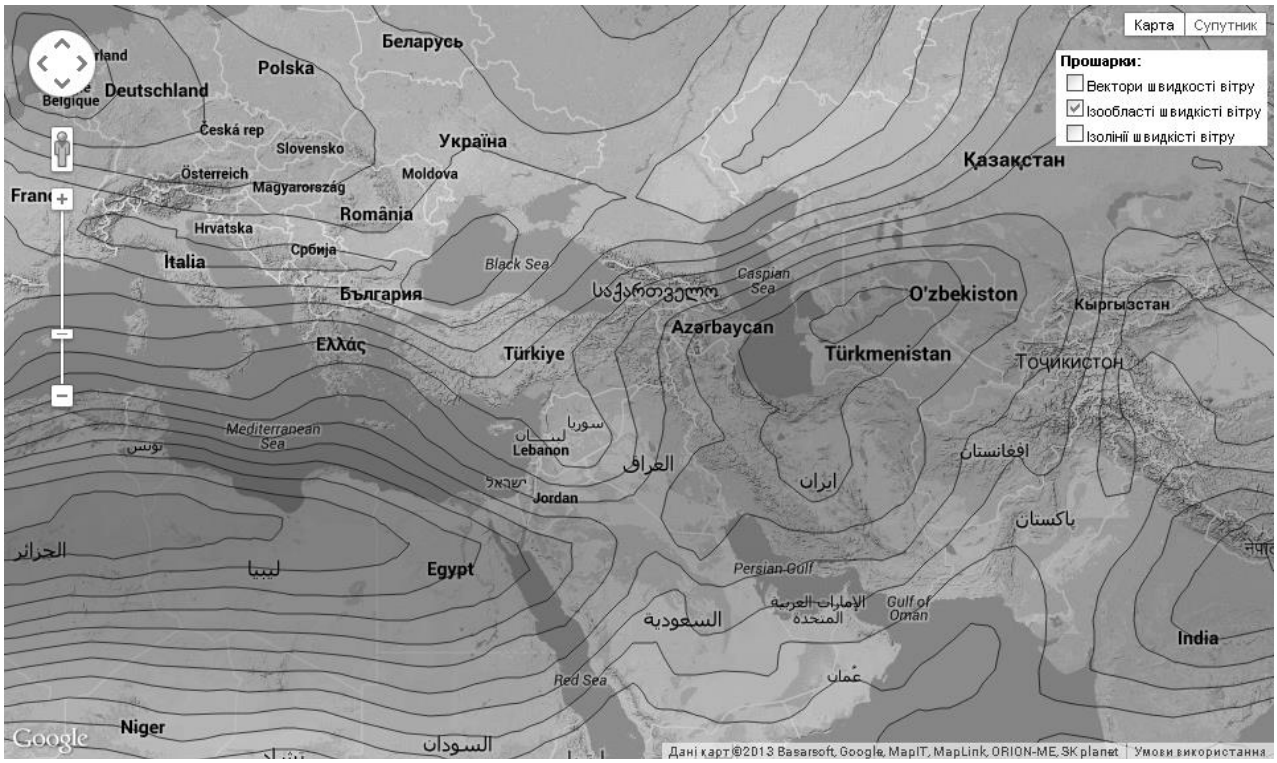

Рис. 9. Картографічне представлення ізообластей вітру

Ці представлення можуть відображатися як окремо так і одночасно.

Для побудови ізоліній використано алгоритм Conrec [11], що є одним з найбільш популярних алгоритмів для обчислення контурних ліній.

Поряд з тим був застосований власноруч розроблений алгоритм для

побудови ізоліній. Далі наведений опис даного алгоритму.

Алгоритм побудови ізоліній. Вхідними даними для алгоритму є прямокутна таблиця розмірами $Mx \times My$ значень деякого параметра z, у даному випадку сили вітру. Представимо її як функцію $z(x, y)$, що визначена на прямокутній області $\Pi = [0, M_x - 1] \times [0, M_y - 1]$, тобто $z(i, j) = z_{i,j}$, $i = 0,..., M_x - 1$, $j = 0,... ..., M_y - 1$.

Множина точок (i, j), $i = 0,... ..., M_x - 1$, $j = 0,..., M_y - 1$, визначає на Π сітку K з вузлами в цих точках. Ребрами сітки K назвемо відрізки, що поєднують вузли сітки. Коміркою сітки K назвемо сукупність кожних чотирьох сусідніх вузлів $(i, j), (i, j+1), (i+1, j+1), (i+1, j)$, які утворюють квадрат.

Нехай тепер потрібно побудувати на Π лінію рівня деякого значення Z функції $z(x, y)$.

Визначимо сітку K структурою, що описує її як масив комірок (рис. 10), розмірністю $[0, M_x - 2] \times [0, M_y - 2]$, елемент якого включатиме у себе: координати кожної вершини комірки, значення параметра в кожній із них та індикатор, що вказує чи проходить крізь дану комірку лінія шуканого рівня.

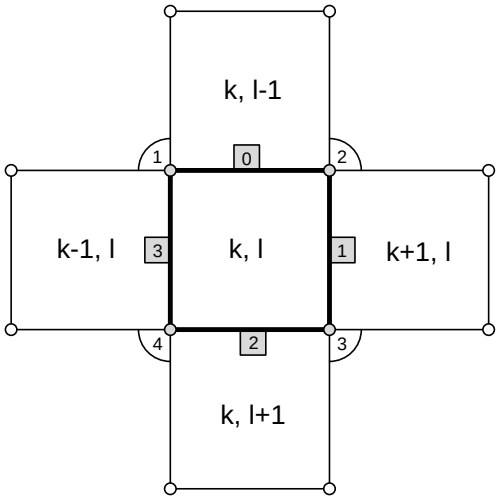

Рис. 10. Комірка, її складові та сусіди, що мають спільні ребра

Для кожної комірки сітки необхідно визначити, чи перетинається вона шуканою ізолінією, для цього порівнюються значення у вузлах, що є вершинами комірки, і з'єднуються ребром. Якщо значення у одному з вузлів виявилось більше Z, а у іншому – менше, то це значить, що лінія рівня перетинає це ребро. Комірка помічається, якщо лінія рівня перетинає хоча б одне з її ребер.

Якщо значення одного з вузлів ребра збігається із Z, то цю ситуацію можна вирішити наступним чином. Введемо поняття початку та кінця ребра. Початком ребра вважатимемо той вузол, сума координат якого менша. Тепер будемо вважати, що лінія перетинає ребро, якщо збігається із Z значення на початку ребра.

Наступним етапом є розрахунок координат точки перетину та поєднання точок у послідовності. Опишемо алгоритм побудови послідовностей.

Крок 1. Послідовно переглядаючи комірки сітки знаходиться комірка, що перетинається лінією шуканого рівня. Перевіряється чи належить знайдена комірка до масиву вже пройдених для лінії шуканого рівня. Вибирається напрямок обходу ребер і згідно напрямку здійснюється пошук будь-якого ребра, що належить до комірки і перетинається із лінією шуканого рівня. Розраховуються координати точки перетину і дана точка ініціює послідовність керуючих точок. Координати точки перетину визначаються за допомогою інтерполяції функції $z(x, y)$ вздовж ребра.

Крок 2. Згідно із напрямком обходу, розглядаються сусідні із поточною комірки на рахунок того, чи перетинаються вони ізолінією. В залежності від місця положення наступної комірки відносно поточної, можна сказати яке ребро поточної комірки перетне ізолінія далі. Розраховуються координати наступної точки і заносяться до масиву послідовності керуючих точок. Пройдені комірки також заносяться до масиву. Комірка, що була знайдена як наступна оголошується поточною.

Крок 3. Якщо є сусідні комірки із помітками і поточна комірка не належить до масиву пройдених, то процедура пошуку наступної точки (крок 2) визивається рекурсивно для даної комірки. Якщо немає сусідніх комірок із перетинами але залишились непройдені комірки перевіряємо чи є остання знайдена комірка сусідньою із першою, якщо так – послідовність оголошується замкненою, якщо ні – незамкненою (рис. 11). Переходимо до кроку 1. Якщо немає сусідніх комірок із перетинами і всі комірки переглянуті, вважається, що ізолінію відслідковано повністю.

Для згладжування форми ізоліній використовується кубічна крива Безьє.

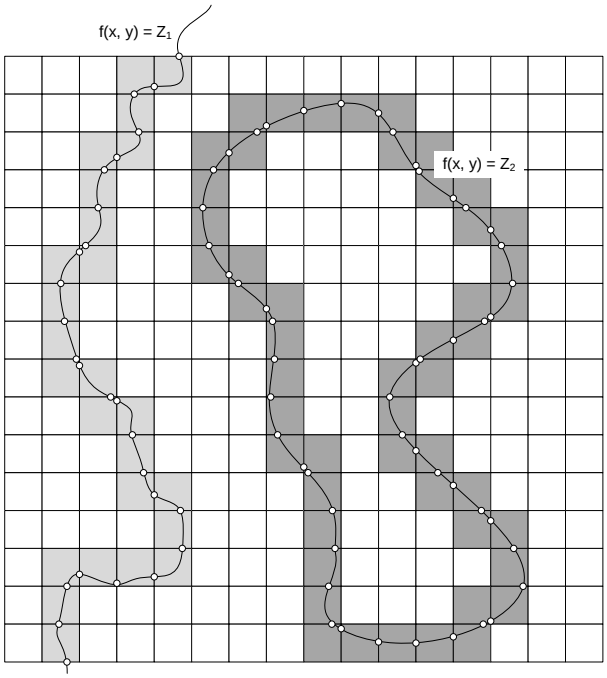

Рис. 11. Замкнена і незамкнена лінії та комірки, які вони перетинають. Відповідні послідовності керуючих точок

Висновки

Розглянуто результати застосування оптимізаційної методології автотьюнінгу до складної наукової задачі. Завдяки розробленому інструментарію значно спрощено і майже повністю автоматизовано етап оптимізації розробленого алгоритму у цільовому обчислювальному середовищі. Досягнутий показник ефективності алгоритму $E(8) = 47.75\%$ можна вважати задовільними, зважаючи на необхідність частої синхронізації проміжних результатів обчислень, що зумовлена характером математичної моделі. Загалом, робота демонструє гнучкість автотьюнінгу й розробленого інструментарію Tuning-Genie, зокрема.

Для візуалізації прогностичних даних моделі розроблено розширення картографічного інструментарію Google Maps, а саме алгоритм генерації ізоліній.

Дана робота розглядає повний цикл розробки програмного забезпечення для вирішення важливої наукової задачі, який вийшло значно спростити за допомогою продемонстрованих інструментальних засобів.

1. *Naono K., Teranishi K., Cavazos J., Suda R.* Software Automatic Tuning From Concepts to State-of-the-Art Results, Springer; 1st edition, September 21, 2010.
2. *Іваненко П.А., Дорошенко А.Ю.* Засоби створення систем автоматичного настроювання для ефективного виконання прикладних паралельних програм // Інформаційні технології в освіті. – 2013. – № 14. – С. 17–21.
3. *Ivanenko P.* TuningGenie – an autotuning framework for optimization of parallel applications // Theoretical and Applied Aspects of Cybernetics. Proceedings of the 2nd International Scientific Conference of Students and Young Scientists. – Kyiv: Bukrek, 2012. – P. 27–30.
4. *Іваненко П.А., Дорошенко А.Ю.* Автоматична оптимізація виконання для задачі метеорологічного прогнозування // Проблеми програмування. – 2012. – № 2–3. – С. 426–434.
5. *Атмосфера*: справочник / [научн. редкол.: Ю.С. Седунов и др.] – Л.: Гидрометеоиздат, 1991. – 601 с.
6. *Черниш Р.І.* Модифіковане адитивно-усереднене розщеплення, його паралельна реалізація та застосування до задач метеорології, дис. канд. техн. наук, Київський Національний університет імені Тараса Шевченка, Київ, 2010 р.
7. *Прусов В.А., Дорошенко А.Е., Черныш Р.И.* Выбор параметра модифицированного аддитивно-усредненного метода // Кибернетика и системный анализ. – 2009. – № 4. – С. 98–105.

8. *Прусов В.А., Дорошенко А.Е., Черныш Р.И.* Метод численного решения многомерной задачи конвективной диффузии // Кибернетика и системный анализ. – 2009. – № 1. – С. 100–107.
9. *http*://maps.google.com/ електронний ресурс.
10. *http*://developers.google.com/maps/documentation/javascript/ електронний ресурс.
11. *http*://paulbourke.net/papers/conrec/ електронний ресурс.

Одержано 12.09.2013

Про авторів:

Іваненко Павло Андрійович,
молодший науковий співробітник,

Дорошенко Анатолій Юхимович,
доктор фізико-математичних наук,
професор, завідуючий відділом
теорії комп'ютерних обчислень,

Овдій Ольга Михайлівна,
молодший науковий співробітник,

Суслова Лідія Миколаївна,
аспірантка.

Місце роботи авторів:

Інститут програмних систем
НАН України
03680, Київ-187,
проспект Академіка Глушкова, 40.
Тел.: (044) 526 1538.
E-mail: paiv@ukr.net,
dor@isofts.kiev.ua,
olga.ovdiy@gmail.com,
2cls87@gmail.com