УДК 681.5.

А.Ю. Дорошенко, П.А. Іваненко, О.С. Новак

ГІБРИДНА МОДЕЛЬ АВТОТЬЮНІНГУ З ВИКОРИСТАННЯМ СТАТИСТИЧНОГО МОДЕЛЮВАННЯ

Розроблена модифікація відомого методу самоналаштування (автотьюнінгу) програм з використанням статистичного моделювання з метою звуження простору пошуку оптимального варіанта програми. Запропонований метод застосовано до оптимізації паралельного гібридного алгоритму сортування. Наведено результати практичного експерименту на мультипроцесорній системі.

Ключові слова: автотьюнінг, статистичне моделювання, автоматизація розробки програмного забезпечення.

Вступ

Питання оптимального використання обчислювальних ресурсів було й залишається актуальним у процесі розробки будь-якого програмного забезпечення – від мобільних додатків до складних клієнт-серверних систем. Парадигма автотьюнінгу [1], яка за останнє десятиріччя стала стандартом для вирішення задачі оптимізації програмних додатків, дозволяє повністю автоматизувати цей процес для будь-якого обчислювального середовища. Її популярність зумовлена в першу чергу простотою застосування й незалежністю від якісних характеристик обчислювача та його операційної системи. Автотьюнінг традиційно використовує емпіричний підхід для отримання якісної оцінки оптимізованої програми (під якістю зазвичай розуміють швидкодію й точність результату). Такий підхід дозволяє уникнути моделювання виконання додатку, що є безумовно складною задачею, а також є точним – важко запропонувати щось більш точне ніж виконання програми у цільовому середовищі. Проте, автотьюнінг будь-яких нетривіальних програмних додатків потребує суттєвих часових затрат. В даній публікації розглядається гібридний підхід до з використанням техніки машинного навчання. Запропонований підхід полягає у автоматичному навчанні моделі на результатах "традиційних" циклів тьюнінгу й подальшій підміні частини запусків програми оцінкою з апроксимаційної моделі.

Методологія автотьюнінгу

Для досягнення максимальної ефективності програми необхідне її додаткове налаштування (тьюнінг) під обчислювальне середовище (ОС), у якому вона буде виконуватися. Як вже було сказано, сучасна методологія самоналаштування (автотьюнінгу) дозволяє автоматизувати це налаштування. Ідея автотьюнінгу полягає у емпіричному оцінюванні декількох варіантів програми й вибору найкращого. Традиційно підбір виконує окрема програма-тюнер, а основними критеріями оцінки є швидкодія й точність отриманих результатів.

Методологія автотьюнінгу активно досліджується у міжнародній науковій спільноті (університети Берклі, Карлсруе та Токіо). Вона вже довела свою універсальність й ефективність проте не позбавлена недоліків, серед яких традиційно виділяють значний час роботи автотьюнера й необхідність його написання.

В цілому дії програми-тюнера є досить шаблонними – обрати/створити нову версію програми й отримати емпіричну оцінку її швидкодії. Тому останній недолік усувається автоматичним генеруванням автотьюнера з вихідного коду оптимізованої програми, що є характеристичною рисою програмної системи автотьюнінгу TuningGenie (детально описана у роботах [2–3]). Завдяки такому підходу вихідна програма займається тільки розв'язанням її основної задачі й

абстрагована від кількісних характеристик ОС, що значно спрощує її розробку. Також така абстракція дозволяє створювати програмні додатки, що будуть автоматично оптимізуватися для виконання на апаратних засобах наступних поколінь (принаймні доки не зміниться багатоядерна архітектура обчислювачів).

Ця стаття висвітлює результати роботи над усуненням головного недоліку автотьюнінгу – істотними часовими затратами, які зумовлені необхідністю емпіричного оцінювання у цільовому середовищі великої множини варіантів вихідної паралельної програми (позначимо множину через C). TuningGenie використовує експертне знання розробника оптимізованої програми для формування C (оформлюється у вигляді мета-даних у вихідному коді), що вже саме по собі мінімізує її розмір. Для додаткового скорочення C) можна застосувати машинне навчання, тобто емпіричну оцінку варіацій $C^{ml} \in C$ пропонується підмінити оцінками з апроксимаційної моделі. Нехай $C^{emp} = | C \setminus C^{ml} |$ – множина варіацій, які будуть оцінюватися емпірично. Насправді маємо двоїсту задачу: $T_{tuning} \to 0$ при $| C^{emp} | \to 0$ й водночас C^{emp} має бути досить великою щоб точно “навчити” модель.

Наступний підрозділ пояснює деталі побудови моделі й дає теоретичну оцінку скорочення часу T_{tuning} .

Модель автотьюнінгу з використанням статистичного навчання

Для початку, визначимо що ми маємо на увазі під терміном «навчання». У широкому розумінні можна вважати що програма отримує досвід (оцінку) *E* стосовно якогось класу задач *T* при замірах продуктивності *P* якщо її продуктивність для задач T оцінених за допомогою P покращується при набутті досвіду E [4].

Машинне навчання тісно пов’язане (і часто перетинається) з обчислювальною статистикою, дисципліною, яка також фокусується на прогнозуванні шляхом застосування комп’ютерів. Воно має тісні зв’язки з математичною оптимізацією, теорією матриць, лінійною алгеброю та копулами, які забезпечують цю галузь методами, теорією та прикладними областями. Машинне навчання іноді з’єднують з добуванням даних [5], де друга підгалузь фокусується більше на дослідницькому аналізі даних, і є відомою як навчання без учителя [6].

Як правило, задачі машинного навчання в залежності від природи «сигналу» поділяють на такі категорії [7]:

- *навчання з учителем*. Системі надаються приклади входів та їхніх бажаних виходів, задані «вчителем», і метою є навчання загального правила, яке відображає входи на виходи;
- *навчання без учителя*. Алгоритмові навчання не дається жодних міток, залишаючи його самому знаходити структуру в своєму вході. Навчання без учителя може бути метою саме по собі (виявлення прихованих закономірностей у даних), або засобом досягнення мети (навчання ознак);
- *навчання з підкріпленням*. Комп’ютерна програма взаємодіє з динамічним середовищем, у якому вона має виконувати певну мету (наприклад, таку як водіння авто) без учителя, який явно казав би їй, чи підійшла вона близько до мети. Іншим прикладом є навчання грі через гру із суперником.

Загалом, можна описати загальний підхід для такого типу задач [8]:

1) *аналіз записів і підготовка «сирих» даних до завантаження*. Оскільки дані можуть приходити з різних джерел і в різних форматах, то потрібно все привести до єдиного вигляду;

2) *підготовка даних*. На цьому етапі вирішуються проблеми неповноти даних, шумів, суперечності в даних и т. п. Основні задачі цього етапу – очищення (заповнення відсутніх значень, видалення спотворених даних та випадкових викидів), перетворення (нормалізація для зниження спотворень), ущільнення (створення виборок даних для окремих атрибутів/груп атрибутів та дискретизація (перетворення неперервних атрибутів в категоріальні);

3) *аналіз даних та побудова моделі;*

4) *перевірка моделі на тестовій виборці даних.*

У контексті нашої задачі автотьюнінгу оціночними параметрами є:

- T – цільова функція автотьюнінгу – швидкодія алгоритму, що вимірюється у (мілі)секундах, оптимізований алгоритм має бути якнайшвидшим;
- P – метрика для числової оцінки параметрів виконання алгоритму, до яких, зокрема, належить кількість паралельних потоків алгоритму;
- E – статистичні спостереження виконання алгоритму, що оптимізується.

Суть методу полягає у зменшенні кількості запусків автотьюнера за допомогою побудови апроксимаційної моделі, що дозволить "відкинути" найменш вірогідні заміри. При цьому апроксимація моделі відбувається завдяки скороченню розмірності вхідних параметрів множини C.

Засоби статистичного моделювання

В подальшому, для статистичного аналізу системи використовувалась мова R. R – мова програмування для статистичних обчислень, аналізу та представлення даних в графічному вигляді [9]. Але вибір інструменту в даному випадку не принциповий, і загалом можна використовувати й інший інструментарій: Python, Julia, S, Matlab і т. д.

Так як при статистичному аналізі ми оперуємо частотними характеристиками, то для їх наглядного відображення зручно використовувати коробковий графік (від англ. "boxplot") [10]. Приклад коробкового графіка показано на рис. 1. Нижня та верхня сторона «ящика» відповідає першому і третьому квантилям, група всередині коробки – другий квантиль (медіана). Кінці «вусів» можуть представляти собою декілька можливих значень – мінімум та максимум даних, граничне значення в межах 1.5 IQR (межквартільний діапазон, від англ. «interquartile range») від нижнього/верхнього квантиля, 1 σ вище/нижче середніх даних, 9-а та 91-а процентилі чи 2-а та 98-а процентилі. Тому варто зауважити, що для всіх далі приведених графіків вуса коробкових графіків відповідають граничним значенням в межах 1.5 IQR від відповідного квантиля.

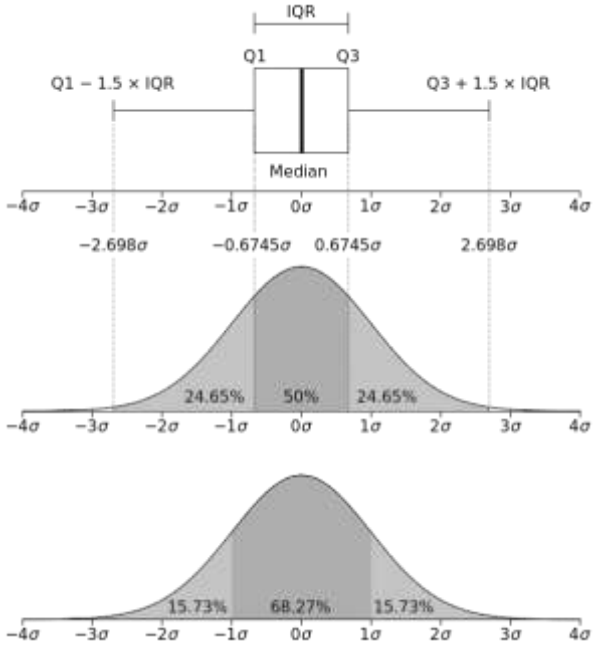

Рис. 1. Приклад коробкового графіка

Практичний експеримент

Як приклад алгоритму для автотьюнінгу було обрано адаптивний алгоритм сортування який виконує сортування злиттям чи вставкою у залежності від розміру блоку числового масиву з даними [9]. Алгоритм був реалізований на мові Java.

Аналіз даних та побудова моделей відбувалися з використанням мови R. R – мова програмування для статистичних обчислень, аналізу та представлення даних в графічному вигляді [10]. Але вибір інструменту в даному випадку не принциповий, і загалом можна використовувати й інший інструментарій: Python, Julia, S, Matlab і т. д.

Сам експеримент складався з декількох стандартних етапів: підготовка та завантаження записів роботи автотьюнера в середовище R, побудова моделі з використанням методів машинного навчання на навчальній вибірці даних та перевірка моделі на контрольній вибірці даних.

Варто підкреслити, що використовувалось два різних набори даних – максимально повний для емпіричного аналізу

системи та мінімально достатній для перевірки моделі.

В рамках даного експерименту виконувалось сортування набору в 10^6 випадкових чисел, а як можливі параметри автотьюнера маємо $C = \{Tcn, Th\}$, де Tcn – кількість потоків виконують обчислення одночасно, Th – розмір блоку при якому буде використовуватися інший алгоритм сортування. Як метрика для оцінки моделі з заданими Tcn та Th використовується T – час виконання алгоритму для Tcn та Th.

При $Tcn = \{2, 4, 6, 8, 10, 12, 14, 16\}$, $Th = [1,800]$ та k =10 очікуваний час виконання автотьюнера $T_{oest} = 6.4 * 10^4 * T_{ex}$, де T_{ex} – час, потрібний на один запуск автотьюнера.

Наведені далі графіки відображають залежність часу виконання цільового алгоритму від вхідних параметрів. Ці данні потрібні на етапі емпіричного аналізу для того, щоб максимально зменшити розмірність системи та скоротити запуски автотьюнера (рис. 2).

В контексті поставленої задачі можна помітити, що вплив кількості потоків оказує на систему більший вплив ніж розмір блоку (рис. 3).

Це можна побачити за наявністю двох явно виражених кластерів даних на першому графіку.

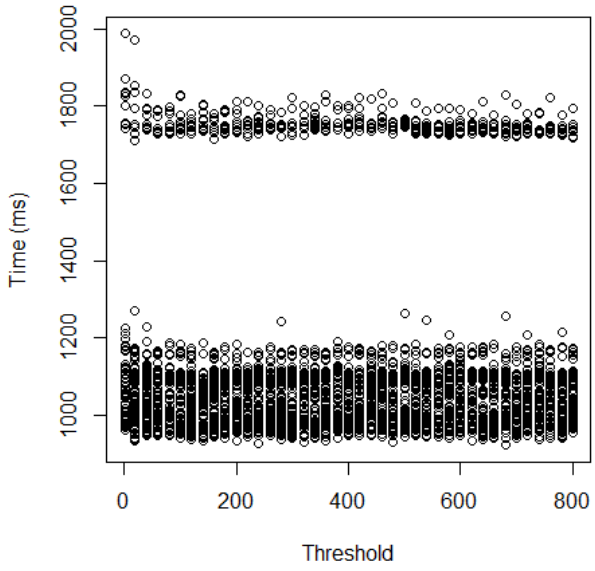

Рис. 2. Вхідні дані моделі

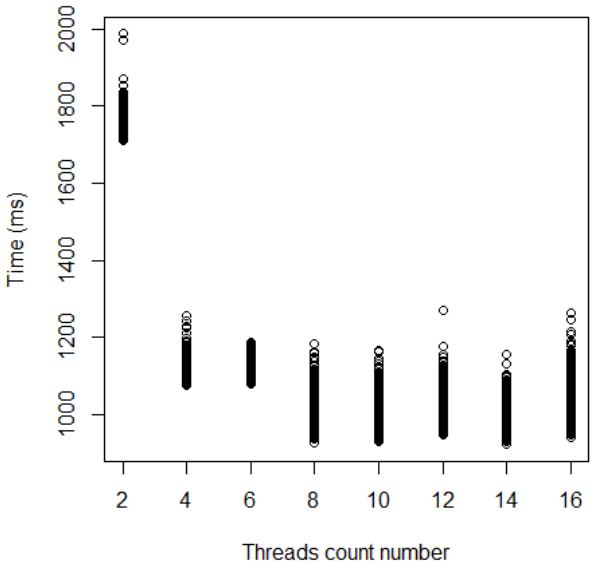

Рис. 3. Частотний розподіл швидкодії за кількістю паралельних потоків

Для швидкої оцінки моделі використовуємо числове інтегрування методом прямокутників (табл. 1). Цей крок дозволить на ранньому етапі відкинути заздалегідь непідходящі варіанти при мінімальній кількості реальних запусків автотьюнера.

Таблиця 1. Наближена оцінка швидкодії

Tcn	2	4	6	8	10	12	14	16
Value	7173 1	4522 5	4526 9	4101 9	4118 4	4135 9	4039 8	4241 9

Як вибіркову функцію використовуємо

$$Tcn_{new} = \{x \in Tcn \mid x < k * \min(Tcn)\},$$

де k – коефіцієнт, що впливає на максимально допустиме відхилення від мінімального значення. При чому, збільшення k значно збільшує час побудови моделі, але при цьому зменшує вірогідність отримати локальний (а не глобальний) мінімум після побудови моделі.

В даному випадку ми звужуємо коло пошуку до $Tcn = \{8, 10, 12, 14, 16\}$.

Оцінивши частотний розподіл після першого спрощення моделі (рис. 4), можна звузити коло пошуку то $Tcn = \{14\}$, що зводить складність задачі з екпоненційної $O(C^n)$ до лінійної $O(n)$ (табл. 2).

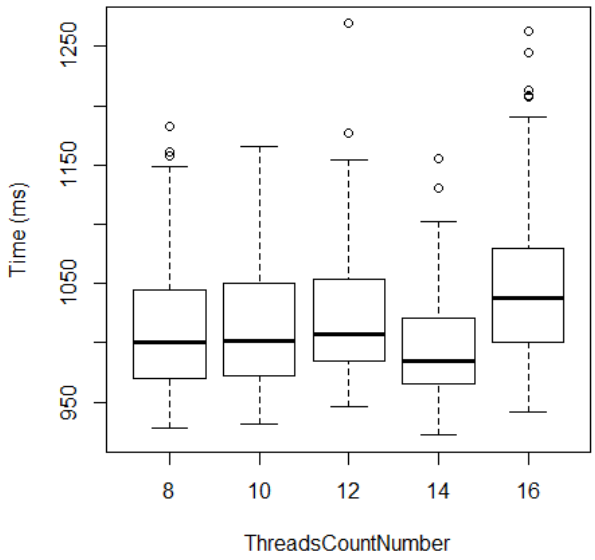

Рис. 4. Частотний розподіл після першого спрощення моделі

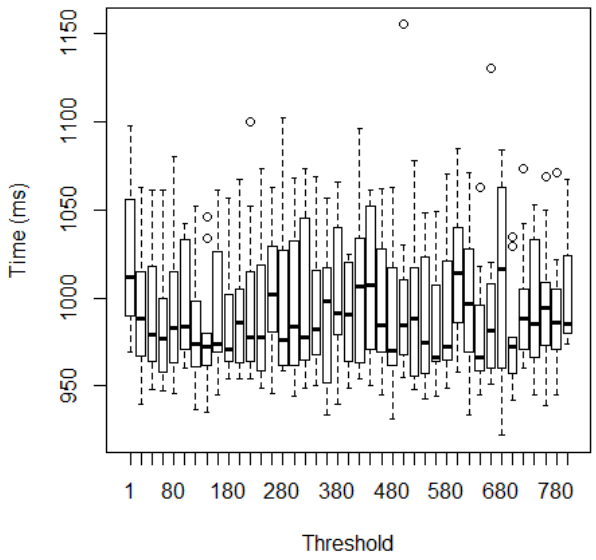

Рис. 5. Остаточний частотний розподіл швидкодії алгоритму

Таблиця 2. Частотний розподіл після першого спрощення моделі

	8	10	12	14	16
Мін	928	931	946	922	942
Нижній квартиль	970	972	984	965	1000
Медіана	1000	1001	1007	984	1037
Верхній квартиль	1044	1050	1054	1021	1079
Макс	1148	1165	1154	1102	1190

Таким чином, виконавши статистичне дослідження системи, стало можливим виконати скорочення розмірності моделі для зменшення часу роботи автотьюнера з

$6.4 * 10^4 * T_{ex}$ до $(8 * 10^2 + k * Tcn) * T_{ex}$,

причому

$$(6.4*10^4 - 8*10^2) \square\ k*Tcn.$$

Практичний експеримент виконувався на комп'ютері з процесором Intel Core i7 2.3 GHz (4 ядра, 256KB L2 Cache, 6MB L3 cache) та 16 GB DDR3@1600 MHz RAM. Оптимальний варіант сортування показав мультипроцесорне прискорення $W(4) = 2.08$ й ефективність $E(4) = 52.08\ \%$ $E(8) = 47.75\ \%$ (рис. 5).

Висновки

В роботі розглянуто гібридний метод автотьюнінгу с використанням статистичного моделювання. Цей метод дозволяє значною мірою позбутися головної слабкості методології автотьюнінгу, а саме суттєво скоротити загальний час пошуку оптимального варіанту програми за рахунок використання статистичної моделі для звуження області пошуку. Для практичного експерименту обрано відносно просту задачу оптимізації паралельного алгоритму сортування чисел. Результати експерименту підтвердили ефективність запропонованого методу й доцільність подальшого розвитку підходу, а саме використання більш складних апроксимаційних функцій та проведення експериментів з паралельними програмами більшої обчислювальної й семантичної складності.

1. *Naono K., Teranishi K., Cavazos J., Suda R.* Software Automatic Tuning From Concepts to State-of-the-Art Results. Springer; 1st Edition. Edition, 2010.
2. *Ivanenko P.A, Doroshenko A.Y., Zhereb K.A.* TuningGenie: Auto-Tuning Framework Based on Rewriting Rules // 10th International Conference, ICTERI 2014, Kherson, Ukraine, June 9-12, 2014, Revised Selected Papers. – P. 139–158.
3. *Иваненко П.А., Дорошенко А.Ю.* Метод автоматической генерации автотюнеров для

параллелных программ // Кибернетика и системный анализ. – 2014. – № 3. – С. 75–83.
4. *Tom M. Mitchell.* Machine learning // mcGraw-Hill Science/Engineering/Math., 1997.
5. *Mannila, Heikki.* Data mining: machine learning, statistics, and databases // Eighth International Conference on Scientific and Statistical Database Systems, 1996.
6. Friedman, Jerome H. Data Mining and Statistics: What's the connection? // Proceedings of the 29th Symposium on the Interface Between Computer Science and Statistics, 1998.
7. *Russell, Stuart, Norvig, Peter.* Artificial Intelligence: A Modern Approach (2nd edition) // Prentice Hall.
8. *Jiawei Han, Micheline Kamber, Jian Pei.* Data mining: Cencepts and Techniques, 3rd edition // Morgan Kaufmann, 2011.
9. *Thomas H.Cormen.* Algorithms unlocked // The MIT Press, 2009.
10. *Michael J.Crawley.* The R Book // John Wiley & Sons Ltd, 2007.

References

1. *Naono K., Teranishi K., Cavazos J., Suda R.* Software Automatic Tuning From Concepts to State-of-the-Art Results. Springer; 1st Edition. Edition, 2010.
2. *Ivanenko P.A, Doroshenko A.Y., Zhereb K.A.* TuningGenie: Auto-Tuning Framework Based on Rewriting Rules // 10th International Conference, ICTERI 2014, Kherson, Ukraine, June 9-12, 2014, Revised Selected Papers. – P. 139–158.
3. *Ivanenko P.A, Doroshenko A.Y.* Method for automated generation of autotuners for parallel applications // Cybernetics and system analisys. – 2014. – N 3. – C. 75–83.
4. *Tom M. Mitchell.* Machine learning // mcGraw-Hill Science/Engineering/Math., 1997.
5. *Mannila, Heikki.* Data mining: machine learning, statistics, and databases // Eighth International Conference on Scientific and Statistical Database Systems, 1996.
6. Friedman, Jerome H. Data Mining and Statistics: What's the connection? // Proceedings of the 29th Symposium on the Interface Between Computer Science and Statistics, 1998.
7. *Russell, Stuart, Norvig, Peter.* Artificial Intelligence: A Modern Approach (2nd edition) // Prentice Hall.
8. *Jiawei Han, Micheline Kamber, Jian Pei.* Data mining: Cencepts and Techniques, 3rd edition // Morgan Kaufmann, 2011.
9. *Thomas H.Cormen.* Algorithms unlocked // The MIT Press, 2009.
10. *Michael J.Crawley.* The R Book // John Wiley & Sons Ltd, 2007.

Одержано 03.10.2016

Про авторів:

Дорошенко Анатолій Юхимович,
доктор фізико-математичних наук,
професор, завідувач відділу
теорії комп'ютерних обчислень,
професор кафедри автоматики та управління в технічних системах НТУУ "КПІ".
Кількість наукових публікацій в
українських виданнях – більше 150.
Кількість наукових публікацій в
іноземних виданнях – понад 30.
Індекс Хірша – 4.
http://orcid.org/0000-0002-8435-1451,

Іваненко Павло Андрійович,
молодший науковий співробітник.
http://orcid.org/0000-0001-5437-9763,

Новак Олександр Сергійович,
аспірант.
http://orcid.org/0000-0002-1665-7360.

Місце роботи авторів:

Інститут програмних систем
НАН України,
03187, м. Київ-187,
проспект Академіка Глушкова, 40.
Тел.: (044) 526 3559.
E-mail: a-y-doroshenko@ukr.net,
paiv@ukr.net,
novak.as@i.ua