УДК 519.5

Ю. Цимбал, О. Пазюк
Національний університет “Львівська політехніка”,
кафедра автоматизованих систем управління

ПОШУК ОПТИМАЛЬНИХ КОМБІНАТОРНИХ СТРУКТУР МЕТОДОМ РОЗПОДІЛЕНИХ ОБЧИСЛЕНЬ

Розглядається проблема пошуку оптимальних комбінаторних структур на прикладі дерев Ліча. Узагальнено критерії оптимальності для таких дерев. Запропоновано застосувати для пошуку систему розподілених обчислень на основі методу повного перебору. Наведено можливі варіанти алгоритмів пошуку, вказано їхні переваги та недоліки.

Ключові слова: комбінаторна структура, дерева Ліча, розподілені обчислення, оптимальні лінійки Ґоломба, розріджені лінійки, масиви Костаса, алгоритм, складність обчислень.

The problem of the search for optimal combinatorial designs on the example of Leech trees has been considered. Criteria of optimality for these trees have been generalized. The application of a system of distributed computing to the search on the basis of exhaustive method has been suggested. Variants of search algorithms have been presented; their advantages and disadvantages have been stated.

Keywords: combinatorial structure, Leech trees, distributed computing, optimum Golomb rulers, Costas set, algorithm, computing complexity.

Вступ

Оптимальні комбінаторні структури [1] стали основою для створення широкого кола сучасних технічних систем. Зокрема, такі структури використовують під час розроблення технологій завадостійкого кодування, у системах захисту інформації, в радіоастрономії та для побудови недвійкових обчислювальних пристроїв [2–5].

Відомими різновидами оптимальних комбінаторних структур є оптимальні лінійки Ґоломба [6, 7], розріджені лінійки [8], масиви Костаса [9], дерева Ліча [10]. Для пошуку лінійок Ґоломба був створений глобальний проект розподілених обчислень OGR [11].

Постановка задачі

Об’єктом нашого дослідження є так звані *дерева Ліча*. Це дерева, ребра яких позначаються натуральними числами, що задають їхню довжину. Відстань між двома вершинами є сумою довжин ребер на шляху між ними. Маркування ребер виконують так, щоб утворити певну неперервну послідовність чисел натурального ряду від 1 до S_1. При цьому дерево буде *обмеженим*, якщо серед відстаней відсутні значення, більші за S_1, і *необмеженим*, якщо присутні (максимальне значення відстані між вершинами позначимо S_2) [2, 3, 10]. Для таких дерев є відомими такі критерії оптимальності.

Оптимальним за Лічем [10] вважається таке дерево, в якому усі відстані між вершинами утворюють *найдовшу* можливу (для даної топології дерева) послідовність чисел натурального ряду від 1 до S_1. Значення відстаней можуть повторюватися. Оптимальні дерева Ліча можуть бути обмеженими і необмеженими (рис. 1, а, б).

Оптимальним за Різником [4, 5] вважається обмежене дерево, в якому множина відстаней, що вичерпують натуральний ряд чисел, відтворюється точно R разів (рис. 1, в). У [4] такі дерева названо “ідеальними розгалуженими лінійками”.

За аналогією з відомою лінійкою Ґоломба, *оптимальним за Ґоломбом* називатимемо дерево Ліча, в якому максимальне значення відстані S_2 є *найменшим* серед можливих для цієї топології. При цьому накладається умова відсутності повторень серед значень відстаней (рис. 1, г).

Ідеальним тоді вважатимемо обмежене дерево Ліча, в якому відстані між вершинами утворюють *найдовшу* можливу (для цієї топології) послідовність чисел натурального ряду без повторень. Очевидно, що ідеальне дерево буде оптимальним за Лічем, Ґоломбом і Різником (при R = 1) одночасно. При цьому $S_1 = S_2 = n \cdot (n+1)/2$, де n – розмірність (кількість ребер) дерева (рис. 1, д).

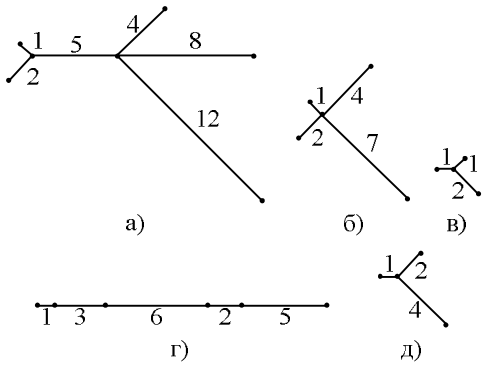

Рис. 1. Приклади дерев Ліча: а – обмежене оптимальне за Лічем (S_1=S_2=20) [10]; б – необмежене оптимальне за Лічем (S_1=9; S_2=11) [10]; в – оптимальне за Різником (S_1=S_2=3; R=2) [4]; г – оптимальне за Ґоломбом (S_1=13; S_2=17 [6]; д – ідеальне (S_1=S_2=6) [10]

Різні за маркуванням дерева однієї топології, які є оптимальними за певним критерієм, утворюють сім’ю оптимальних дерев.

Метою досліджень є створення програмної системи для пошуку дерев Ліча, які будуть оптимальними за вказаними критеріями для усіх можливих топологій різної розмірності.

Аналіз методів досліджень

Сьогодні відомими є методи пошуку лише для деяких різновидів дерев Ліча. Для знаходження лінійок Ґоломба, зокрема, використовується апарат циклічних різницевих множин [12, 13]. Для лінійок, оптимальних за Лічем, можливий спосіб побудови запропонував Віхман [14, 8]. Проте, на нашу думку, доволі складно розробити прості та універсальні методи знаходження оптимальних дерев довільної топології, зважаючи на стрімке зростання кількості таких топологій зі збільшенням кількості вершин [15].

Отже, як метод пошуку дерев Ліча пропонується використати повний перебір варіантів маркування для всіх можливих топологій заданої розмірності. До позитивних рис такого підходу слід віднести можливість гарантованого знаходження повних сімей оптимальних дерев, одночасної перевірки дерева за усіма критеріями оптимальності та достатньо простої побудови алгоритмів перебору і перевірки. Недоліком є відома проблема “комбінаторного вибуху”, що пов’язана з експоненційним зростанням обсягів обчислень при збільшенні кількості вершин у дереві.

Маркувати дерево певної топології пропонується так. Задається значення суми всіх довжин ребер S, яке розбивається усіма можливими способами на натуральні доданки (довжини відповідних ребер). Оскільки порядок доданків є важливим, задача зводиться до генерації *композицій* числа S [16]. Мінімальне значення S дорівнює n. Максимальне значення суми (надалі позначатимемо його S_3) для дерева довільної топології оцінити доволі складно. Відомо, що для лінійок Ґоломба $S_3 = S_2 \approx n^2$ (для великих n) [17]. Для розгалужених дерев, оптимальних за Лічем,

існує лише приблизна оцінка $S_2 < (n+1)^2/2$ [10]. Припускається, що значення S_3 для дерев, оптимальних за Ґоломбом, буде співвимірним з відповідним значенням для нерозгалуженої лінійки.

Приблизно оцінімо складність обчислень, задавши емпірично значення $S_3 = n^2$. Тоді при маркуванні певного дерева загальна кількість генерованих композицій N_c дорівнює $\sum_{S=n}^{n^2} C_{S-1}^{n-1}$. У табл. 1 наведено значення кількості композицій як можливих способів маркування дерев з кількістю ребер від 3 до 10.

Таблиця 1

Кількість композицій як можливих способів маркування дерев

Кількість ребер, n	Кількість топологій дерев, T_n	Кількість композицій для однієї топології, N_c	Загальна кількість композицій, $T_n \times N_c$
3	2	84	168
4	3	1 820	5 460
5	6	53 130	318 780
6	11	1 947 792	21 425 712
7	23	85 900 584	1 975 713 432
8	47	4 426 165 368	208 029 772 296
9	106	260 887 834 350	27 654 110 441 100
10	235	17 310 309 456 440	4 067 922 722 263 400

Аналіз результатів обчислень показав, що зростання кількості композицій залежно від збільшення числа ребер має яскраво виражений експоненційний характер.

Алгоритм пошуку дерев Ліча

З вищевикладеного випливає, що пошук оптимальних дерев Ліча у запропонований спосіб на одному комп'ютері для значень n, які перевищують 8, є неприйнятним з часових міркувань. Отже, для пришвидшення пошуку за методом повного перебору доцільно розробити та застосувати систему розподілених обчислень [18]. Структурну схему такої системи наведено на рис. 2.

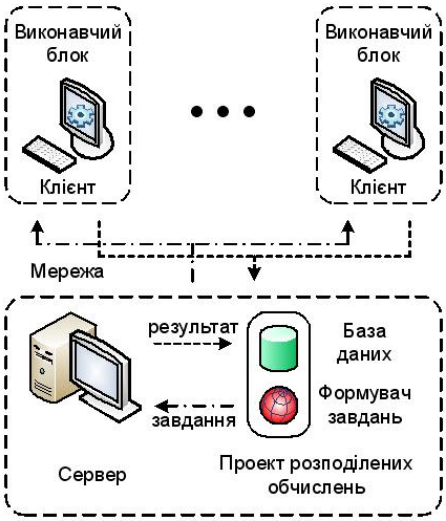

Рис. 2. Структурна схема системи розподілених обчислень

Передусім треба розробити алгоритм пошуку, тобто визначити, як саме розподілити між сервером та клієнтами задачі формування дерев-кандидатів, визначення їхніх характеристик (S_1, S_2, R) та перевірки на оптимальність. Формування повного набору дерев-кандидатів полягає у

маркуванні ребер усіма можливими способами. Для цього можна застосувати різні комбінаторні алгоритми [19–21]. Ми випробували 4 варіанти розподілу задач. Спільним для них є зовнішній цикл перебору значень суми довжин ребер S від n до S_3, який виконує сервер і в якому формуються завдання для клієнтів. Також в усіх варіантах клієнти визначають оптимальні дерева-кандидати в межах отриманого завдання (рис. 3).

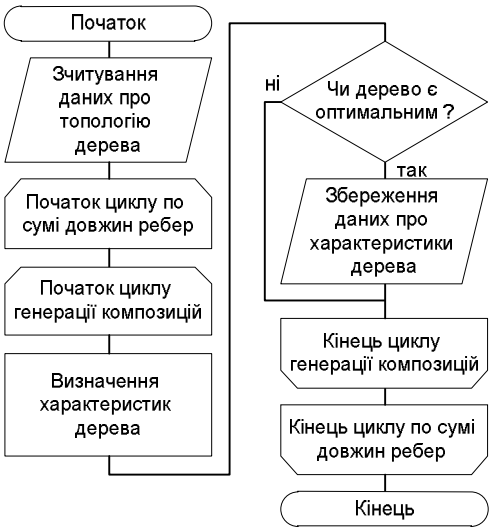

Рис. 3. Узагальнена блок-схема алгоритму пошуку оптимальних дерев Ліча

За *першим* варіантом розподілу задач *сервер* генерує всі *можливі розбиття* кожної суми, а *клієнт* виконує всі *можливі перестановки* елементів розбиття (повторення елементів допускаються).

Такий варіант дає змогу розбити весь обсяг обчислень на достатньо дрібні завдання і, відповідно, швидко і з малими витратами пам'яті опрацювати їх клієнтами. Проте при цьому час формування завдань на сервері набагато перевищує час обробки, а для зберігання проміжних результатів на сервері треба задіяти значний об'єм пам'яті.

Другий варіант покладає генерацію *розбиттів* сум та *перестановок* елементів на *клієнта*.

При цьому розвантажується серверна частина, але із збільшенням розмірності задачі стрімко зростатиме час опрацювання завдання і обсяг потрібної пам'яті на клієнтах. Також зазначимо, що навіть у межах однієї топології дерева розмір і час опрацювання завдання для різних значень суми істотно відрізнятимуться.

У *третьому* варіанті замість почергової генерації розбиттів і перестановок застосовано генерацію *композицій* (усіх можливих перестановок для усіх можливих розбиттів у межах одного алгоритму), яку виконує *клієнт*.

За обсягом і розподілом задіяної пам'яті цей варіант є близьким до попереднього, але безпосередня генерація композицій дає змогу пришвидшити обчислення на клієнті.

Четвертий варіант, на відміну від попереднього, передбачає розподіл задачі генерації *композицій* для кожного значення S поміж *різними клієнтами*. При цьому всі клієнти отримають приблизно однакові завдання, розмір яких можна гнучко налаштувати на сервері.

Потенційною перевагою цього варіанта є збалансовування навантаження на клієнтів та пов'язаних витрат часу, уникнення можливого переповнення пам'яті на сервері та клієнті.

Для всіх варіантів одним зі способів вирішення проблем, що пов'язані з пам'яттю, є своєчасне збереження проміжних і кінцевих результатів у базі даних.

Наведені варіанти у найпростішій реалізації не передбачають обміну інформацією між клієнтами і надсилання до клієнтів проміжних результатів. Отже, потрібно провести *додаткове опрацювання* отриманого від клієнтів переліку дерев-кандидатів і визначити оптимальні для

опрацьованої топології. Іншою задачею буде виявлення *ізоморфних* дерев серед оптимальних і їх вилучення.

Як програмну платформу для створення системи розподілених обчислень обрано ***hxGrid*** (автор – Роман Лут) [22]. ***hxGrid*** написано мовою програмування ***Delphi***, працює на різних версіях ОС ***Windows***. За його допомогою можна відносно легко і швидко розгорнути проект розподілених обчислень. ***hxGrid*** – вільно розповсюджуваний продукт з відкритими текстами програмних модулів, що дає змогу змінювати код клієнтської та серверної частин, гнучко налаштовувати інтерфейс та конфігурувати параметри системи.

Висновки

Розроблено тестову модель системи розподілених обчислень для пошуку оптимальних дерев Ліча, яка реалізує різні варіанти розподілу задач між сервером і клієнтами. Вказані вище переваги і недоліки є попередніми і потребують перевірки в реальних умовах для задач різної розмірності і для різної кількості клієнтів. Наступною першочерговою задачею є реалізація і застосування системи для дерев малої розмірності ($n \leq 8$) та здійснення ґрунтовного аналізу витрат часу та пам'яті. Це дасть змогу обрати доцільний варіант розподілу задач для опрацювання дерев більшої розмірності, а також виявляти та узагальнювати певні закономірності серед отриманих оптимальних дерев різних топологій.

1. Colbourn C.J., Dinitz J.H. Handbook of Combinatorial Designs (2 ed.). – Chapman & Hall - CRC, 2007. – 1018 p. 2. Bloom G.S., Golomb S.W. Applications of Numbered Undirected Graphs // Proceedings of the IEEE. – 1977. – Vol. 65, No. 4, pp. 562-570. 3. Bloom G.S., Golomb S.W. Numbered Complete Graphs, Unusual Rulers, and Assorted Applications // Theory and Applications of Graphs. – Springer, 1978, pp. 53-65. 4. Різник В.В. Синтез оптимальних комбінаторних систем. – Львів: Вища школа. Вид-во при Львів. ун-ті, 1989. – 168 с. 5. Наукова школа професора Володимира Різника http://iknit.lp.edu.ua/riznyk. 6. Golomb Ruler http://mathworld.wolfram.com/GolombRuler.html. 7. Линейка Голомба http://ru.wikipedia.org/wiki/Линейка_Голомба. 8. Luschny P. Perfect And Optimal Rulers http://www.luschny.de/math/rulers/introe.html. 9. Costas J.P. A Study of a Class of Detection Waveforms Having Nearly Ideal Range-Doppler Ambiguity Properties // Proceedings of the IEEE. – 1984. – Vol. 72, No. 8, pp. 996-1009. 10. Leech J. Another Tree Labelling Problem // The American Mathematical Monthly. – 1975. – Vol. 82, No. 9, pp. 923-925. 11. distributed.net: Project OGRhttp://www.distributed.net/ogr/. 12. Dimitromanolakis A. Analysis of the Golomb Ruler and the Sidon Set Problems, and Determination of Large, Near-Optimal Golomb Rulers. – Thesis. Department of Electronic and Computer Engineering. Technical University of Crete, June 2002. – 118 p. 13. Drakakis K. A Review of the Available Construction Methods for Golomb Rulers // Advances in Mathematics of Communications. – 2009. – Vol. 3, No. 3, pp. 235-250. 14. Wichmann B. A Note on Restricted Difference Bases // Journal of the London Mathematical Society. – 1963. – Vol. 38, pp. 465-466. 15. Rosen K.H. (ed.). Handbook of Discrete and Combinatorial Mathematics. – CRC Press, 2000. – 1183 p. 16. Knuth D.E. The Art of Computer Programming. Volume 4 Fascicle 3, Generating All Combinations and Partitions. – Addison-Wesley, 2005. – 156 p. 17. Atkinson M.D., Santoro N., Urrutia J. Integer Sets with Distinct Sums and Differences and Carrier Frequency Assignments for Nonlinear Repeaters // IEEE Transactions on Communications. – 1986. – Vol. COM-34, No. 6, pp. 614-617. 18. Пазюк О.В., Цимбал Ю.В. Система розподілених обчислень для пошуку оптимальних числових лінійок // Комп'ютерні науки та інформаційні технології: Матеріали 4-ї Міжнародної науково-технічної конференції CSIT-2009. – Львів: Вид-во ПП "Вежа і КО", 2009. – с. 214-216. 19. Ruskey F. Combinatorial Generation http://www.1stworks.com/ref/RuskeyCombGen.pdf. 20. Kreher D.L., Stinson D.R. Combinatorial Algorithms. Generation, Enumeration, and Search. CRC Press, 1999. – 340 p. 21. Skiena S.S. The Algorithm Design Manual, 2nd Edition. – Springer, 2008. – 739 p. 22. Проект hxGrid http://www.deep-shadows.com/hax/hxgrid.htm