УДК 004.8

Н.Р. Білоус
В.О. Денисюк

ДОСЛІДЖЕННЯ ТА РЕАЛІЗАЦІЯ ПАРАЛЕЛЬНОГО АЛГОРИТМУ ПОШУКУ BREADTH-FIRST SEARCH ЗА ДОПОМОГОЮ OPENMP

Вінницький національний технічний університет

Анотація

У роботі розглянуто розробку та програмну реалізацію паралельного алгоритму пошуку в ширину Breadth-First Search (BFS), орієнтованого на обробку великих графових структур у середовищі спільної пам'яті. Актуальність теми зумовлена широким застосуванням обходу графів у задачах аналізу соціальних мереж, пошуку найкоротших шляхів у незважених графах, маршрутизації, побудови дерев досяжності, а також у багатьох прикладних напрямах обробки великих даних. BFS є базовим алгоритмом, який формує рівневу структуру обходу, що створює природні передумови до паралельного виконання: вершини одного рівня можуть опрацьовуватися незалежно з подальшим переходом до наступного рівня. У межах дослідження проаналізовано принципи рівневого (frontier-based) виконання BFS, визначено основні проблеми паралелізації, зокрема конфлікти доступу до спільних структур даних (позначення відвіданих вершин, масиви рівнів та предків), а також накладні витрати синхронізації між потоками. Для реалізації паралельного алгоритму використано технологію OpenMP, що забезпечує зручні засоби організації багатопотокових обчислень на багатоядерних процесорах. Програмний модуль реалізовано мовою C++ із представленням графа у вигляді списків суміжності, що є ефективним для розріджених графів та знижує витрати пам'яті порівняно з матрицею суміжності. Проведено експериментальне тестування послідовної та паралельної реалізацій BFS при різній кількості потоків. Визначено показники часу виконання, прискорення та ефективності, а також досліджено вплив збільшення кількості потоків на масштабованість. Отримані результати демонструють, що паралельна реалізація здатна зменшувати час обробки, проте при надмірному збільшенні кількості потоків можливе зниження ефективності через конкуренцію за ресурси пам'яті та витрати на синхронізацію. Практичне значення роботи полягає у можливості використання отриманих підходів для прискорення графових алгоритмів у задачах, де критичним є час обробки великих обсягів даних.

Ключові слова: *Breadth-First Search, паралельний алгоритм, OpenMP, обхід графа, фронт обходу, продуктивність, прискорення, ефективність.*

Abstract

The paper considers the development and software implementation of a parallel Breadth-First Search (BFS) algorithm designed for processing large graph structures in a shared-memory environment. The relevance of the topic is driven by the widespread use of graph traversal in practical tasks such as social network analysis, shortest path search in unweighted graphs, routing, reachability tree construction, and various big data processing pipelines. BFS is a fundamental algorithm that builds a level-by-level traversal structure, which naturally enables parallel execution: vertices belonging to the same level can be processed independently before moving to the next level. Within the study, the principles of level-synchronous (frontier-based) BFS execution are analyzed and the main parallelization challenges are identified. In particular, special attention is paid to data races and shared-state conflicts when updating common data structures (visited markers, level arrays, and parent links), as well as to synchronization overhead between threads. The parallel implementation is developed using OpenMP, which provides convenient constructs for organizing multithreaded computations on multicore CPUs. The software module is implemented in C++, and the graph is represented by adjacency lists, which is efficient for sparse graphs and reduces memory consumption compared to adjacency matrices. A set of experiments is carried out to compare sequential and parallel BFS versions for different thread counts. The execution time, speedup, and parallel efficiency metrics are evaluated, and the scalability behavior is investigated. The obtained results show that the parallel approach can reduce processing time; however, excessive thread counts may lead to efficiency degradation due to increased synchronization costs and contention for memory

bandwidth. The practical value of the work lies in providing an applicable approach to accelerate graph algorithms in scenarios where the processing time of large-scale datasets is critical.

Keywords: *Breadth-First Search, parallel algorithm, OpenMP, graph traversal, frontier, performance, speedup, efficiency.*

Вступ

Сучасні інформаційні системи дедалі частіше оперують великими обсягами даних, структура яких природним чином описується у вигляді графів. Графові моделі широко застосовуються в аналізі соціальних мереж, системах рекомендацій, задачах маршрутизації та навігації, комп'ютерних мережах, біоінформатиці, а також у задачах пошуку та обробки знань. У таких умовах особливого значення набувають ефективні алгоритми обходу графів, здатні забезпечити прийнятний час виконання при значній кількості вершин і ребер.

Одним із базових алгоритмів обходу графів є пошук у ширину (Breadth-First Search, BFS), який здійснює поетапний рівневий обхід графа, починаючи з заданої стартової вершини. Алгоритм BFS використовується для знаходження найкоротших шляхів у незважених графах, побудови дерев досяжності, визначення компонент зв'язності та розв'язання багатьох інших прикладних задач. Класична послідовна реалізація BFS має часову складність порядку O(V+E), де V — кількість вершин, а E — кількість ребер графа. Проте при роботі з великими графовими структурами навіть така лінійна складність може виявитися недостатньо ефективною з точки зору практичного часу виконання.

Зростання обчислювальних можливостей сучасних процесорів відбувається переважно за рахунок збільшення кількості обчислювальних ядер, що зумовлює необхідність використання паралельних обчислень для підвищення продуктивності програмних систем. У цьому контексті актуальним є дослідження можливостей паралельної реалізації алгоритмів обходу графів, зокрема BFS. Рівнева організація алгоритму BFS створює природні передумови для паралелізації, оскільки вершини, що належать одному рівню обходу, не мають прямих залежностей між собою і можуть опрацьовуватися незалежно.

Разом із тим паралельна реалізація BFS супроводжується низкою практичних проблем, серед яких ключовими є синхронізація потоків, запобігання конфліктам доступу до спільних структур даних та ефективне управління пам'яттю. Вибір засобів паралелізації відіграє важливу роль у досягненні балансу між простотою реалізації та продуктивністю. Однією з найбільш поширених технологій для організації паралельних обчислень у середовищі спільної пам'яті є OpenMP, яка надає зручні директиви для розпаралелювання циклів і керування виконанням потоків без суттєвого ускладнення програмного коду.

Актуальність даного дослідження полягає у розробці та аналізі паралельної реалізації алгоритму Breadth-First Search із використанням OpenMP, а також у практичній оцінці її ефективності на багатоядерних процесорах. Отримані результати дозволяють краще зрозуміти особливості масштабованості BFS у середовищі спільної пам'яті та можуть бути використані як основа для подальших досліджень і оптимізації паралельних графових алгоритмів.

Постановка задачі дослідження

Метою даної роботи є розробка та дослідження паралельної реалізації алгоритму пошуку в ширину Breadth- First Search у середовищі спільної пам'яті з використанням технології OpenMP, а також оцінка ефективності такого підходу при обробці графів великої розмірності.

- Для досягнення поставленої мети в роботі необхідно розв'язати такі задачі:
- проаналізувати теоретичні основи алгоритму Breadth-First Search та особливості його послідовної реалізації;
- дослідити можливості паралелізації BFS з урахуванням його рівневої організації та принципу формування фронтів обходу;
- визначити основні проблеми паралельної реалізації BFS, зокрема конфлікти доступу до спільних структур даних та накладні витрати на синхронізацію потоків;

- обґрунтувати вибір технології OpenMP як засобу реалізації багатопотокових обчислень у середовищі спільної пам’яті;
- розробити програмну реалізацію послідовного та паралельного варіантів алгоритму BFS мовою C++;
- провести експериментальні дослідження продуктивності реалізованих алгоритмів при різній кількості потоків;
- виконати аналіз отриманих результатів, оцінити показники прискорення та ефективності, а також дослідити масштабованість паралельної реалізації.

Розв’язання зазначених задач дозволяє комплексно оцінити доцільність використання паралельного підходу для алгоритму BFS та визначити практичні обмеження його застосування у багатоядерних обчислювальних системах.

Виклад основного матеріалу

Алгоритм пошуку в ширину (Breadth-First Search, BFS) є фундаментальним алгоритмом теорії графів, який використовується для обходу графових структур рівень за рівнем, починаючи з заданої стартової вершини. Він знаходить широке застосування у задачах аналізу мереж, пошуку найкоротших шляхів у незважених графах, моделюванні соціальних і комунікаційних мереж, а також у багатьох інформаційних системах, де необхідна обробка великих графових даних [1-3]. Основною перевагою BFS є гарантоване знаходження мінімальної відстані в кількості ребер, однак ця властивість досягається за рахунок значної кількості операцій доступу до пам’яті.

Класична послідовна реалізація BFS базується на використанні черги, в якій зберігаються вершини поточного фронту обходу. Алгоритм поступово розширює фронт, відвідуючи всі суміжні вершини, що ще не були оброблені, та присвоюючи їм відповідні рівні. При зростанні розміру графа до сотень тисяч або мільйонів вершин послідовна реалізація BFS демонструє значне зростання часу виконання, що обумовлено як обсягом обчислень, так і нерегулярним доступом до оперативної пам’яті.

З огляду на широке поширення багатоядерних процесорів актуальним є дослідження можливостей паралельної реалізації BFS. Проте паралелізація цього алгоритму є нетривіальною, оскільки BFS має рівневу структуру, а кількість вершин у кожному рівні може істотно змінюватися. Це призводить до нерівномірного розподілу навантаження між потоками та обмежує масштабованість алгоритму [4, 5].

У даній роботі для реалізації паралельного BFS було обрано модель паралельних обчислень зі спільною пам’яттю на основі технології OpenMP. OpenMP надає зручний інструментарій для створення паралельних регіонів, розподілу циклів між потоками та синхронізації доступу до спільних ресурсів, що робить його придатним для експериментального дослідження паралельних алгоритмів на CPU [4, 6-10]. Паралельна реалізація BFS, розроблена в межах цієї роботи, базується на концепції фронту обходу. На кожному кроці алгоритму всі вершини поточного рівня формують фронт, який розподіляється між потоками. Кожен потік незалежно обробляє підмножину вершин фронту та виконує перегляд їхніх суміжних вершин. Для формування наступного фронту використовується допоміжна структура, в яку потоки додають нові вершини, що були відвідані вперше.

Ключовою проблемою при такій паралелізації є забезпечення коректного доступу до спільних масивів рівнів та батьківських вершин. Оскільки кілька потоків можуть одночасно намагатися відвідати одну й ту саму вершину, у реалізації застосовано механізм критичних секцій OpenMP. Це дозволяє уникнути конфліктів доступу та забезпечує коректність результатів, проте водночас призводить до додаткових накладних витрат, пов’язаних із синхронізацією потоків.

Для оцінки продуктивності реалізованого алгоритму було проведено серію експериментів на графах різного розміру: 10 000, 100 000, 1 000 000 та 10 000 000 вершин. Графи генерувалися програмно та мали структуру лінійного ланцюжка з додатковими «стрибковими» ребрами, що дозволяє змоделювати реальні графові структури з поєднанням локальних та віддалених зв’язків. Такий підхід забезпечує більш репрезентативну оцінку поведінки BFS у практичних задачах.

Перший етап тестування було проведено на мобільній платформі з процесором AMD Ryzen 7 5800H та оперативною пам’яттю DDR4-3200. Результати вимірювання часу виконання наведені в таблиці 1.

Таблиця 1 – Результати тестування програми на платформі 1 (ms)

Розмір графу	Послідовно	Паралельно, 1 потік	Паралельно, 2 потоки	Паралельно, 4 потоки	Паралельно, 8 потоків
10 000	5.840	3.530	4.537	6.223	9.111
100 000	56.327	31.663	31.707	36.958	53.171
1 000 000	565.937	315.213	316.19	348.912	492.580
10 000 000	5 599.461	3 160.957	3 137.687	4 130.215	5 902.432

Аналіз отриманих даних показує, що для невеликих графів ефект від паралелізації є мінімальним. Це пояснюється тим, що накладні витрати на створення паралельних регіонів і синхронізацію потоків переважають виграш від паралельного виконання. Для графів великого розміру, зокрема для 1 000 000 і 10 000 000 вершин, паралельна версія BFS демонструє суттєве скорочення часу виконання порівняно з послідовною реалізацією. Найкращі результати досягаються при використанні 2–4 потоків.

Подальше збільшення кількості потоків до 8 не приводить до пропорційного зростання продуктивності. У деяких випадках спостерігається навіть погіршення результатів, що зумовлено конкуренцією потоків за доступ до оперативної пам’яті та зростанням витрат на синхронізацію. Це характерно для memory-bound алгоритмів, де продуктивність обмежується пропускною здатністю пам’яті, а не обчислювальними ресурсами CPU [4].

Другий етап експериментів було виконано на настільній високопродуктивній платформі з процесором AMD Ryzen 9 9900X та оперативною пам’яттю DDR5-6000. Результати тестування наведені в таблиці 2.

Таблиця 2 – Результати тестування програми на платформі 2 (ms)

Розмір графу	Послідовно	Паралельно, 1 потік	Паралельно, 2 потоки	Паралельно, 4 потоки	Паралельно, 8 потоків
10 000	3.202	1.423	1.873	2.692	6.390
100 000	31.196	15.107	16.529	18.999	44.503
1 000 000	304.053	180.124	173.730	180.579	478.906
10 000 000	2 979.283	1 786.913	1 696.304	1 648.938	3 750.874

Отримані результати демонструють суттєве зменшення часу виконання BFS для всіх розмірів графів порівняно з мобільною платформою. Це пояснюється вищою тактовою частотою процесора, більшою кількістю ядер, збільшеним кешем третього рівня та значно вищою пропускною здатністю оперативної пам’яті. Проте, незважаючи на кращі абсолютні показники продуктивності, характер масштабованості алгоритму залишається подібним: оптимальний діапазон кількості потоків становить [2, 3, 6].

Таким чином, результати експериментів підтверджують, що структура алгоритму BFS накладає фундаментальні обмеження на його масштабованість. Зі збільшенням кількості потоків після певної межі накладні витрати на синхронізацію, роботу зі спільними структурами даних та нерегулярний доступ до пам’яті починають переважати виграш від паралельного виконання. Це узгоджується з відомими теоретичними та практичними результатами досліджень паралельних графових алгоритмів Незалежно від апаратної платформи оптимальним є використання 2–4 потоків, що забезпечує найкращий баланс між прискоренням та накладними витратами. Отримані результати підтверджують доцільність застосування OpenMP для паралельної реалізації BFS у задачах аналізу великих графових структур.

Висновки

У роботі досліджено алгоритм пошуку в ширину (Breadth-First Search) як один із базових алгоритмів теорії графів та проаналізовано особливості його послідовної і паралельної реалізації. Розглянуто принципи роботи алгоритму BFS, його властивості та обмеження при обробці великих графових структур. На основі аналізу сучасних джерел досліджено можливості паралелізації BFS у середовищі зі спільною пам'яттю з використанням технології OpenMP.

Програмно реалізовано послідовну та паралельну версії алгоритму BFS мовою програмування C++. Проведено експериментальне тестування розробленої програми на графах різного розміру та на різних апаратних платформах. Проаналізовано отримані результати, зокрема час виконання, коефіцієнт прискорення та коефіцієнт ефективності паралельної реалізації.

Встановлено, що паралельна реалізація BFS забезпечує помітне прискорення для великих графів, однак ефективність паралелізації обмежується накладними витратами на синхронізацію та доступ до пам'яті. Показано, що оптимальною кількістю потоків у більшості випадків є 2–4, тоді як подальше збільшення кількості потоків не приводить до зростання продуктивності та може знижувати ефективність алгоритму.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Ахо А., Хопкрофт Д., Ульман Дж. Структури даних та алгоритми. Київ: Видавнича група BHV, 2002. 536 с.
2. Кормен Т., Лейзерсон Ч., Рівест Р., Штайн К. Алгоритми. Побудова та аналіз. 3-є видання. Київ: Видавнича група BHV, 2013. 1296 с.
3. C++ Довідкова документація. URL: https :// w 3 schoolsua . github . io / cpp / cpp _ ref _ reference . html # gsc.tab =0 .
4. Grama A., Gupta A., Karypis G., Kumar V. Introduction to Parallel Computing. 2nd ed. Addison-Wesley, 2003. 856 p.
5. Voss M. Parallel Graph Algorithms with OpenMP. In: OpenMP Shared Memory Parallel Programming. Springer, 2008. pp. 129–140.
6. Quinn M. J. Parallel Programming in C with MPI and OpenMP. McGraw Hill, 2003. 529 p.
7. Chapman B., Jost G., van der Pas R. Using OpenMP: Portable Shared Memory Parallel Programming. MIT Press, 2007. 400 p.
8. Методичні вказівки до лабораторних робіт з навчальної дисципліни «Паралельні та розподілені обчислення» (частина 1) для здобувачів вищої освіти першого (бакалаврського) рівня за освітньопрофесійною програмою «Комп'ютерна інженерія» спеціальності 123 «Комп'ютерна інженерія» денної і заочної форм навчання [Електронне видання] / Бойчура М. В., Шатний С. В., Шатна А. В. Рівне : НУВГП, 2023. 49 с.
9. Проектування та аналіз обчислювальних алгоритмів: Вступ до алгоритмів [Електронний ресурс]: навчальний посібник для студ. спеціальності 122 «Комп'ютерні науки»/ І. В. Федорін; КПІ ім. Ігоря Сікорського. Електронні текстові дані (1 файл: 1,97 Мбайт). Київ: КПІ ім. Ігоря Сікорського, 2022. 115 с.
10. Skiena S. The Algorithm Design Manual. Springer, 2020. 758 p.

Білоус Назар Русланович – студент кафедри комп`ютерних наук, факультет інтелектуальних інформаційних технологій та автоматизації, Вінницький національний технічний університет, м.Вінниця, e-mail: ya.belous.nazar@gmail.com;

Денисюк Валерій Олександрович – канд. техн. наук, доцент, доцент кафедри комп'ютерних наук, Вінницький національний технічний університет, м.Вінниця, e-mail: vad64@i.ua.

Bilous Nazar Ruslanovich – student of Computer Science Department, Faculty of Intelligent Information Technologies and Automation, Vinnytsia National Technical University, Vinnytsia, e-mail: ya.belous.nazar@gmail.com;

Denysiuk Valerii Olexandrovich – Ph.D., Assistant Professor, Assistant Professor of the Chair of Computer Science, Vinnytsia National Technical University, Vinnytsia, e-mail: vad64@i.ua .