УДК 004.04

С.Д. Погорілий, д-р техн. наук, професор,
А.В. Потебня, студент
Київський національний університет імені Тараса Шевченка, м. Київ, Україна
sdp77@i.ua, poteba@yandex.ru

Формування та дослідження паралельної схеми алгоритму Крускала для систем зі спільною пам'яттю

Виконано формалізацію алгоритму Крускала побудови мінімального покривного дерева неорієнтованого графа з використанням математичного апарату модифікованих систем алгоритмічних алгебр В.М. Глушкова. Запропоновано концепцію його розпаралелювання для архітектур комп'ютерних систем зі спільною пам'яттю. Проведено низку досліджень алгоритму та сформовано його часові характеристики. Наведено рекомендації щодо використання алгоритму при розв'язанні прикладних задач.

Ключові слова: архітектури зі спільною пам'яттю, мінімальне покривне дерево, алгоритм Крускала, потік, системи алгоритмічних алгебр, РСА, ПРСА, еквівалентні перетворення схем, розпаралелювання, граф

Вступ

Надзвичайно поширеними у багатьох сферах сучасного життя є моделі зважених графів, у яких деяке значення ваги (weight) або вартості (cost) пов'язане з кожним ребром [1]. Відповідно, розв'язання складних теоретичних та прикладних задач, які при цьому виникають, здійснюється за рахунок досягнень графової алгоритмізації. Наприклад, при розробці надвеликих інтегральних схем (VLSI), виникає необхідність з'єднати їх компоненти у такий спосіб, щоб одержаний пристрій був компактним, споживав мало потужності та швидко проводив сигнали. Подібні проблеми постають при розробці мережі доріг, а також при розв'язанні задач кластеризації та плануванні руху робота [2].

Покривним деревом (spanning tree) неорієнтованого графа називається ациклічний підграф, що пов'язує всі його вершини та, відповідно, є деревом [3]. Під мінімальним покривним деревом (МПД), відповідно, розумітимемо покривне дерево, вага якого є найменшою [1]. Дане поняття використано в працях багатьох вчених. Наприклад, у роботах [4, 5] запропоновано підхід до розв'язання задачі маршрутизації для бездротових сенсорних мереж з використанням алгоритму побудови МПД. Зокрема, відсутність циклів у сформованому дереві дозволяє уникнути випадку так званого «широкомовного шторму». Варто зауважити також про використання МПД в космології для вивчення статистичних властивостей вибірок галактик [6].

Алгоритм Крускала вигідно використовувати при знаходженні мінімального покривного дерева для розріджених графів. Варто зауважити, що на відміну від алгоритму Прима, час виконання алгоритму Крускала залежить не лише від кількості вершин графа, але й від його насиченості (тобто, кількості ребер).

Метою дослідження є формування регулярної схеми алгоритму (РСА) та паралельної регулярної схеми алгоритму (ПРСА) за допомогою математичного апарату модифікованих систем алгоритмічних алгебр (САА-М). Виконано їх експериментальну реалізацію для систем зі спільною пам'яттю з метою дослідження прискорення паралельного алгоритму та встановлення залежностей часу роботи алгоритмів від різних наборів вхідних даних.

Алгоритм Крускала

Нехай, граф G задано за допомогою множини вершин V, множини ребер E та вагової функції $\omega: E \to R$. Побудова МПД починається з виродженого лісу, що складається з |V| дерев, кожне з яких містить по одній вершині [1]. Потім послідовно серед усіх ребер, додавання яких до МПД не спричинить появу в ньому циклу, обирається ребро мінімальної ваги та включається до МПД. Важливим кроком алгоритму є сортування ребер за вагою. Для цього використовуються базові алгоритмічні конструкції. Наприклад, у цій роботі обрано спосіб швидкого сортування.

Слід зазначити, що згідно з [1], існують два способи завершення алгоритму Крускала. Тут виберемо спосіб, детальний опис якого подано в [3]. Крім того, скористаємося евристиками об'єднання за рангом (union by rank) та стиснення шляху (path compression). Ідея першої з них

полягає у використанні для кожного кореня рангу, що являє собою верхню границю висоти вузла. Відповідно, при об'єднанні корінь з меншим рангом повинен вказувати на корінь з більшим рангом. Згідно з другою евристикою кожен вузол повинен посилатися безпосередньо на корінь свого дерева [3].

Таким чином, схема алгоритму Крускала має вигляд:

створити пусту множину МПД;
для кожної вершини графа $u \in V$
повторювати
створити відповідну компоненту зв'язаності;
кінець циклу
відсортувати ребра графа за вагою в неспадному порядку;
для кожного ребра $(u,v) \in E$ у порядку зростання їх ваги
повторювати
якщо вершини ребра належать до різних дерев
то
додати ребро (u,v) до МПД;
сполучити дерева;
повернути МПД;

Розглянемо приклад виконання цього алгоритму. Нехай, задано граф, зображений на рис. 1а. Відсортована послідовність ребер має вигляд:

Ребро	Вага ребра
2 – 4	5
1 – 3	7
1 – 4	9
1 – 2	10
3 – 5	12
1 – 5	14
4 – 5	20

На початку роботи алгоритму маємо ліс, що складається з 5 піддерев МПД, кожне з яких містить по одній вершині. На рис. 1б – 1г ребра найменшої ваги послідовно долучаються до МПД, а відповідні піддерева об'єднуються. При додаванні ребра 1 – 2 утворюється цикл, тому воно відкидається (рис. 1д). Подібні міркування застосовуються і при роботі з іншими ребрами.

Результат виконання алгоритму наведено на рис. 1е. Виділені ребра належать до МПД. Сумарна вага отриманого дерева становить 33.

Системи алгоритмічних алгебр

Відомо, що алгеброю є множина елементів з визначеною на ній системою (сигнатурою) операцій – функцій, що приймають значення на цій множині. В кібернетиці до числа найбільш фундаментальних належить поняття зворотного зв'язку, на якому засноване управління функціонуванням складних систем різноманітної природи. Зворотний зв'язок втілено в концепції абстрактної моделі керування, що базується на абстрактній моделі Глушкова однопроцесорної ЕОМ. Для формалізованого представлення алгоритмів функціонування абстрактної моделі ЕОМ В. М. Глушков запропонував математичний апарат систем алгоритмічних (мікропрограмних) алгебр (САА) [10].

Перевагою цього апарату порівняно з класичними алгоритмічними системами, такими, як машини Тьюринга, рекурсивні функції і т. п., є його орієнтація на проектування алгоритмів та асоційованих з ними програм.

Фіксована САА представляє собою двоосновну алгебраїчну систему, основами якої є множина операторів та множина умов. Операції САА поділяються на логічні та операторні. До логічних відносяться узагальнені булеві операції та операція лівого множення оператора на умову (призначена для прогнозування обчислювального процесу), а до операторних – основні конструкції структурного програмування (послідовне виконання, розгалуження та циклічне повторення).

Фундаментальне значення має теорема Глушкова, згідно з якою для довільного алгоритму існує (в загальному випадку, не єдина) САА, в якій цей алгоритм може бути представлено регулярною схемою.

З абстрактними моделями багатопроцесорних обчислювальних систем асоційовані модифіковані САА, отримані шляхом введення трьох додаткових операцій, орієнтованих на формалізацію паралельних обчислень. До них належать, зокрема, фільтрація $\underline{\alpha}$, синхронна диз'юнкція $A \vee B$ та асинхронна диз'юнкція $A \dot{\vee} B$. Більш докладно ці операції описано в [10].

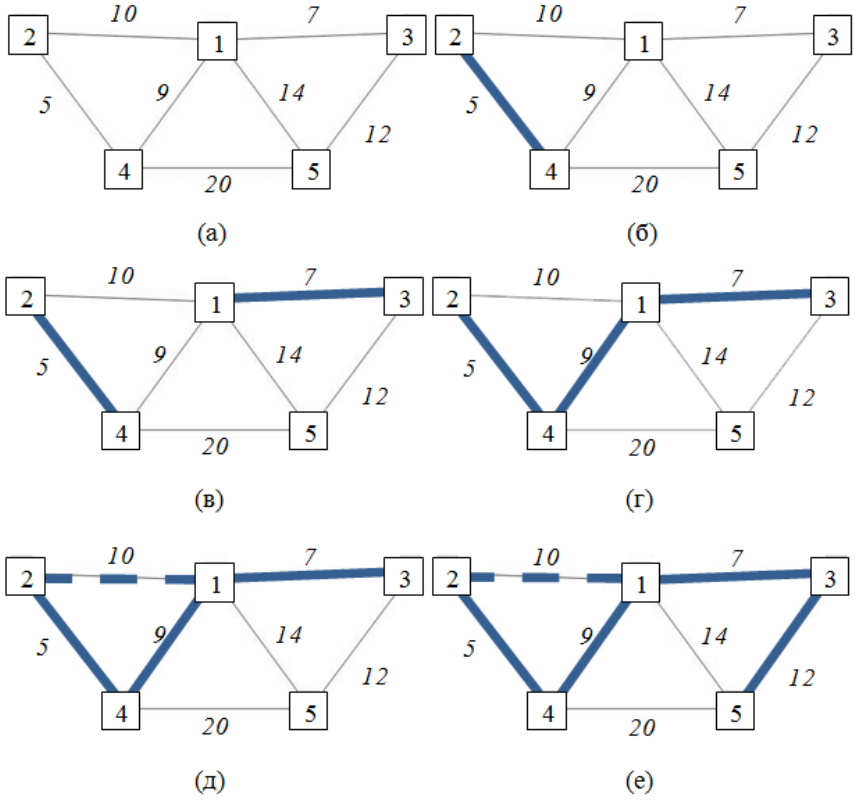

Рисунок 1 – Приклад роботи алгоритму Крускала

Формалізація алгоритму Крускала

Формалізуємо алгоритм Крускала шляхом формування схеми з використанням апарату САА Глушкова. Як було згадано вище, для цього треба визначити певну множину операторів та умов.

Введемо позначення для операторів:

M – створення порожньої множини МПД;

K_i – формування компоненти зв'язаності, що відповідає і-й вершині графа;

Q = QuickSort(E) – швидке сортування множини ребер графа E;

U_j – об'єднання піддерев, що містять вершини j-го ребра множини E;

A_j – додавання j-го ребра до МПД.

Множина предикатів:

$i \le |V| \Rightarrow \alpha$ – і менше або дорівнює кількості вершин графа;

$j \le |E| \Rightarrow \beta$ – j менше або дорівнює кількості ребер графа;

$(E[j].u \in K_m) \wedge (E[j].v \in K_l) \wedge (m \neq l) \Rightarrow \gamma$ – вершини ребра належать до різних піддерев.

Таким чином, схема алгоритму може бути записана у такому вигляді:

$$M * \{K_j\} * Q * \{(U_j * A_j, E)\}$$
$$\alpha \qquad \beta\gamma$$

Одержане співвідношення є послідовною РСА алгоритму Крускала.

Формування паралельної регулярної схеми алгоритму Крускала

Відомо, що маючи САА схему алгоритму Крускала, можна шляхом застосування апарату еквівалентних перетворень трансформувати послідовний алгоритм в паралельний [10]. Застосувавши властивість розпаралелювання альтернативи $(A, E) = \underset{\alpha}{A}$ та позначивши за $U \Rightarrow U_j * A_j$, маємо:

$$M * \{K_j\} * Q * \{\underset{\beta}{\gamma} U\}$$
$$\alpha$$

Слід зазначити, що фільтри забезпечують гнучке керування обчислювальним процесом. При цьому розрахунки по гілках з істинними фільтруючими умовами продовжуються, а інші гілки розриваються.

Встановлено, що найбільш громіздкою операцією є сортування списку ребер. Ідея розпаралелювання швидкого сортування полягає в розподілі вхідних даних між потоками. Варто зауважити про можливість нерівномірного переділу навантаження, що призводитиме до збільшення сумарного часу блокування потоків

та, відповідно, до затримок в роботі програми. Позначивши сортування окремих частин списку за допомогою операторів $Q_1, Q_2, ..., Q_n$, маємо:

$$M * \underset{\alpha}{\{ K_j \}} * Q_1 * Q_2 * ... * Q_n * \underset{\beta}{\{ \underline{\gamma} U \}}$$

Пристосувавши до отриманої формули властивість $\underset{\alpha}{\{ A * B \}} = \underset{\alpha}{\{ A \}} \dot{\vee} \underset{\alpha}{\{ B \}}$, виконаємо розпаралелювання швидкого сортування. Отримаємо:

$$M * \underset{\alpha}{\{ K_j \}} * \# \overset{n}{\underset{k=1}{\dot{\bigvee}}} Q_k \# * \underset{\beta}{\{ \underline{\gamma} U \}}$$

Символами «#...#» позначається частина, оператори якої виконуються паралельно, а символами «$\dot{\vee}$» – оператори асинхронної диз'юнкції.

До останньої формули пристосуємо розподільчу властивість паралельного виконання $A * \# B \dot{\vee} C \# = \# A * B \dot{\vee} A * C \#$, розподіливши формування лісу піддерев МПД. Для зручності позначимо за $K \Rightarrow \underset{\alpha}{\{ K_j \}}$. Маємо:

$$M * \# \overset{n}{\underset{k=1}{\dot{\bigvee}}} (K * Q_k) \# * \underset{\beta}{\{ \underline{\gamma} U \}}$$

Отримана схема представляє ПРСА алгоритму Крускала.

Системи зі спільною пам'яттю

Відомо, що основною вимогою при удосконаленні будь-якого алгоритму є зменшення часу його виконання. Пошук вирішення цієї проблеми призвів до розробки паралельних обчислювальних систем [7]. Згідно з класифікацією Фліна вони поділяються на 4 класи [11]. Важливе місце посідають системи зі спільною пам'яттю (Symetric Multiprocessing, SMP). До таких систем відносять багатоядерні комп'ютери та мультипроцесори. Їх характерною рисою є те, що всі процесори мають прямий та рівноправний доступ до будь-якої точки спільної пам'яті. Наявність спільної пам'яті значно полегшує організацію взаємодії процесорів один з одним, відповідно, спрощуючи програмування. Утім, ця архітектура виявилася непридатною для створення масштабних систем у зв'язку з виникненням великої кількості конфліктів при звертанні до спільної шини [8].

Попри те, що SMP системи не можуть конкурувати з можливостями потужних кластерів, ця архітектура є найбільш розповсюдженою серед робочих станцій та серверів. Тому сучасні операційні системи сімейств Windows NT, UNIX та Linux мають вбудовану підтримку режиму SMP [9].

Експериментальна реалізація паралельної схеми алгоритму Крускала для систем зі спільною пам'яттю

Для дослідження алгоритму використано апаратну архітектуру AMD Athlon II P320 – 2.1 GHz, 2 ядра. Програмні реалізації створені мовою C# на платформі .NET Framework 4. Розпаралелювання алгоритму здійснено за допомогою потоків.

Відомо, що для розподілу виконання різних застосувань в операційних системах використовуються процеси. Всередині процесу може виконуватися один або більше потоків. Слід зазначити, що порівняно з реалізацією прикладної програми у вигляді кількох взаємодіючих процесів багатопотокова модель має ряд переваг:

1. Створення (як і завершення) нового потоку в межах процесу відбувається набагато швидше, ніж створення (завершення) нового процесу;

2. Перемикання потоків в межах процесу виконується швидше, ніж перемикання процесів;

3. Ефективність обміну інформацією між потоками одного процесу підвищується за рахунок можливості прямого доступу до спільних ресурсів без посередництва операційної системи [9].

Все ж, варто зазначити, що для будь-яких потоків, що працюють з одними і тими самими даними, неможливо наперед вказати їх відносні швидкості виконання, а отже, результат їх роботи залежить від випадкових чинників. Таке явище має назву гонок (race condition).

Попри невдачі в ранніх версіях, що були пов'язані з необхідністю керування потоками на низькому рівні, розробники .NET Framework 4 досягли значних успіхів у реалізації засобів забезпечення багатопотоковості шляхом надання нових бібліотек класу, середовища виконання та діагностики [12]. Варто зауважити, що кількість потоків може змінюватися при виконанні таких програмних конструкцій. Рекомендації щодо їх використання при розробці паралельних програм сформовано в роботі [13].

На рис. 2 наведено залежності часу виконання послідовного та паралельного алгоритмів від розмірності графа. Форма цих кривих спричинена поліноміальною складністю швидкого сортування. На підтвердження цього в таблиці 1 продемонстровано вплив часу сортування на загальний час роботи алгоритму Крускала.

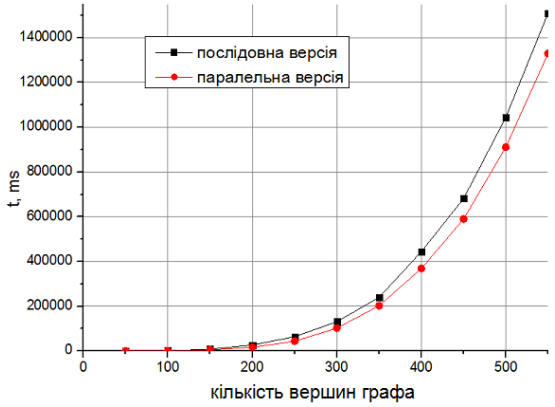

Рисунок 2 – Залежність часу виконання алгоритму від розмірності графа

Таблиця 1 – Вплив часу сортування на час роботи алгоритму

|V|	частка сортування, %	
	послідовна версія	паралельна версія
50	97,0740	93,6310
100	99,2281	97,7682
150	99,6500	99,3831
200	99,7969	99,6573
250	99,8724	99,7892
300	99,9116	99,8770
350	99,9338	99,9198
400	99,9539	99,9387
450	99,9596	99,9489
500	99,9598	99,9601
550	99,9730	99,9686

На рис. 3 подано залежність відношення часу послідовної версії алгоритму до часу виконання паралельної. Для з'ясування причин виникнення піку на отриманій характеристиці, розглянемо архітектуру використаного процесора. Кожне його ядро має власну виділену кеш-пам'ять першого та другого рівнів. При «кеш-попаданні» читання (read hit) дані вибираються з локального кеша. При «кеш-промаху» читання (read miss) дані вибираються з основної пам'яті та заносяться у локальний кеш [7]. У зв'язку з обмеженим обсягом кеш-пам'яті L2, при збільшенні розмірності графа, зростатиме кількість «кеш-промахів», тобто послідовних операцій звертання до оперативної пам'яті, що й призводить до появи відповідного піку на характеристиці прискорення.

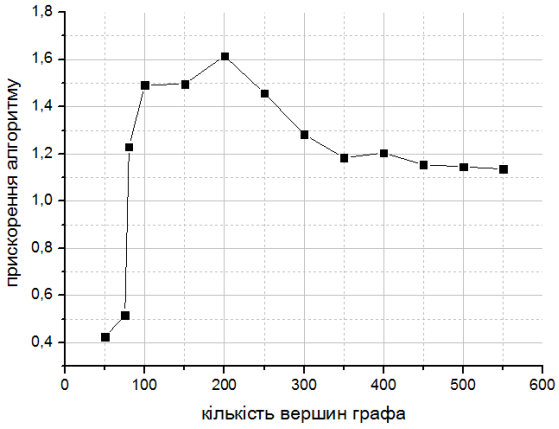

Рисунок 3 – Залежність відношення часу виконання послідовної версії алгоритму до часу паралельної

На рис. 4 сформовано залежність часу виконання обох алгоритмів від кількості вершин графа та його насиченості (відношення кількості наявних ребер до їх максимально можливої кількості). Отримані результати підтверджують доцільність використання алгоритму Крускала на розріджених графах.

Цілком природно виграш від використання паралельної версії зростає зі збільшенням насиченості графа, що визначається обсягом роботи швидкого сортування.

Варто зауважити, що час роботи алгоритму залежить також від кількості можливих значень вагової функції графа. Вигляд цієї залежності подано на рис. 5. Очевидно, що у випадку лише одного можливого значення паралельний алгоритм працюватиме набагато повільніше у зв'язку відсутністю переваг від паралелізму, а, натомість, наявністю витрат, пов'язаних зі створенням та знищенням потоків.

Для більш детального дослідження цієї залежності розглянемо її окремий випадок для графа, що містить 150 вершин (рис. 6). Як з'ясувалося, форма кривої для послідовного алгоритму визначається загальною кількістю операцій перестановки ребер при здійсненні сортування (рис. 7). На рис. 8 подано залежність сумарного часу блокування потоків від кількості значень вагової функції графа. З її вигляду можна зробити висновок про вплив блокування на результуючий час роботи алгоритму.

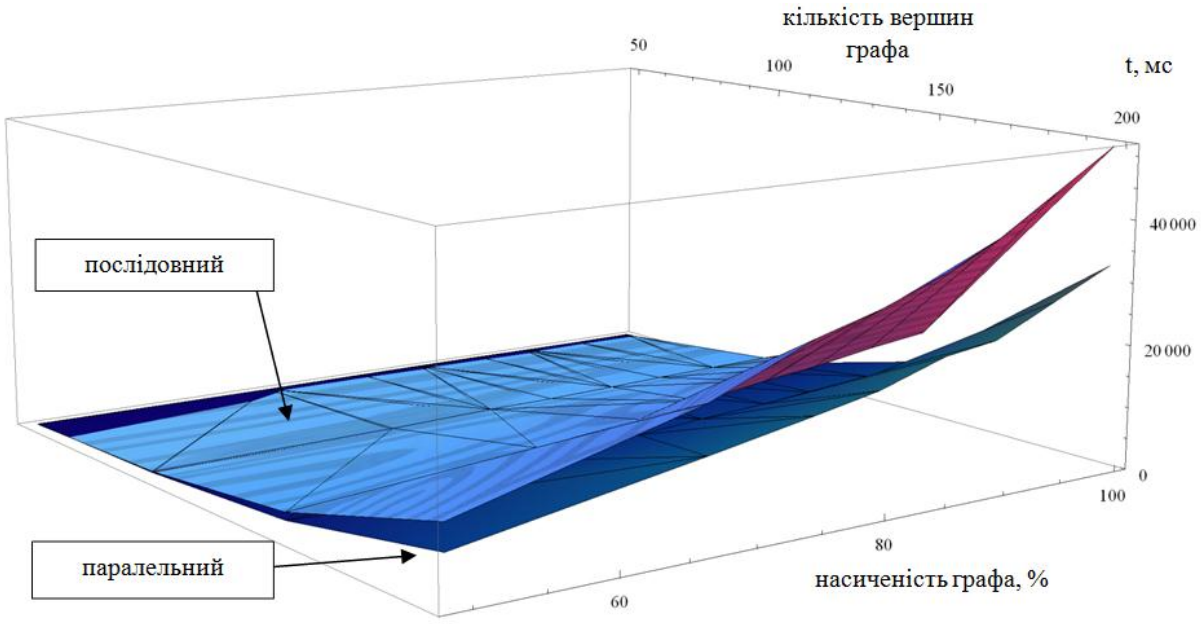

Рисунок 4 – Залежність часу виконання послідовного та паралельного алгоритмів від кількості вершин графа та його насиченості

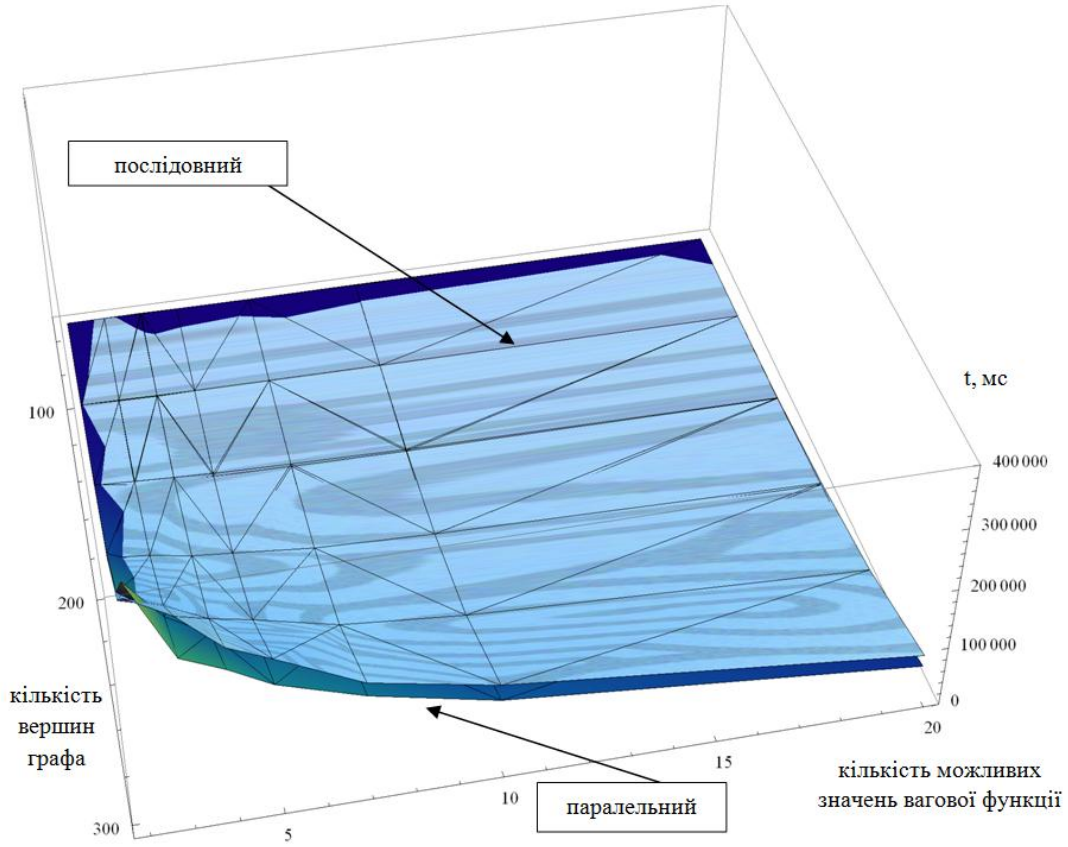

Рисунок 5 – Залежність часу виконання послідовного та паралельного алгоритмів від кількості можливих значень вагової функції графа

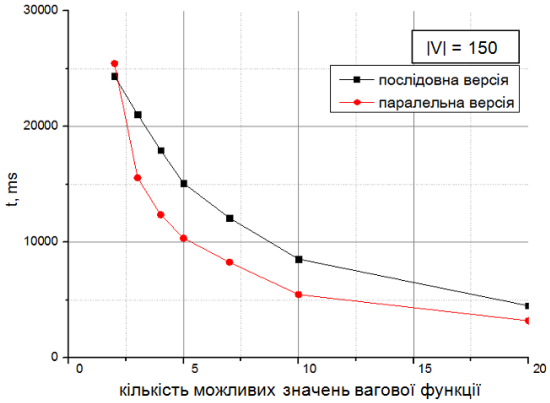

Рисунок 6 – Залежність часу виконання алгоритму від кількості можливих значень вагової функції графа (у випадку 150 вершин)

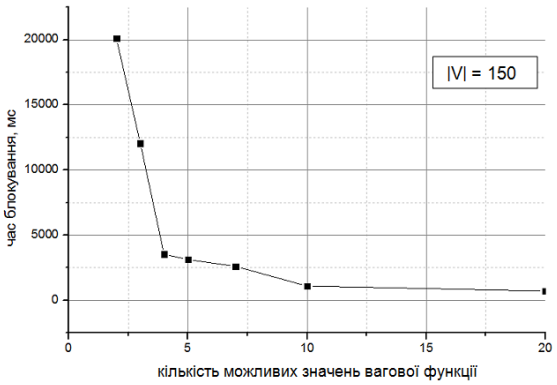

Рисунок 8 – Залежність сумарного часу блокування потоків від кількості можливих значень вагової функції

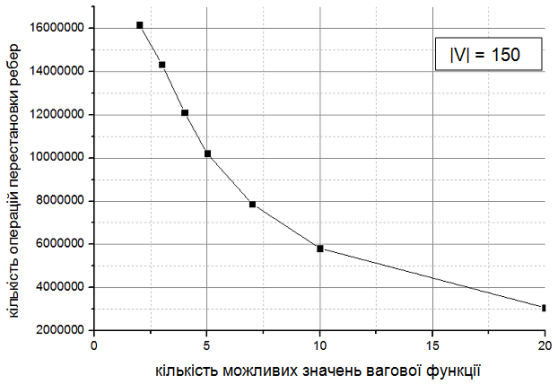

Рисунок 7 – Відстеження кількості операцій перестановки ребер

Висновки

У статті виконано формалізацію алгоритму Крукала з використанням математичного апарату CAA-M, отримано його PCA і ПРСА. Запропоновано концепцію розпаралелювання алгоритму для обчислювальних систем зі спільною пам'яттю. Варто зауважити, що при розробці паралельної програми найкращі показники продуктивності продемонстрували конструкції паралельних циклів. Отримана залежність прискорення алгоритму від розмірності графа має пік, що пояснюється обмеженістю ресурсу кеш-пам'яті. Дослідження експериментальної реалізації отриманих схем підтверджують доцільність використання алгоритму Крускала при формуванні МПД для розріджених графів. Встановлено та обґрунтовано залежність часу роботи обох версій алгоритму від кількості можливих значень вагової функції графа.

Подальший напрямок досліджень пов'язаний з реалізацією та дослідженням паралельної версії алгоритму Крускала для систем з розподіленою пам'яттю.

Список літератури

1. Седжвик Р. Фундаментальные алгоритмы на C++. Алгоритмы на графах / Р. Седжвик; пер. с англ. – СПб.: ООО «ДиаСофтЮП», 2002. – 496 с.
2. David Eppstein Spanning Trees and Spanners / David Eppstein // University of California, Irvine, CA 92717. – 35 p.
3. Алгоритмы. Построение и анализ / Томас Кормен, Чарльз Лейзерсон, Рональд Ривест, Клиффорд Штайн. – М. :Издательский дом «Вильямс», 2005. – 1296 с.
4. Денисенко В.С. Модификация алгоритма построения кратчайшего остовного дерева для беспроводной сенсорной сети / В.С. Денисенко, А.В. Шостак // Системи обробки інформації. – 2010. – № 2(83). – С. 75 – 77.
5. Gagarin A. Distributed Search for Balanced Energy Consumption Spanning Trees in Wireless Sensor Networks / Gagarin A., Hussain, S., Yang, L.T. // Advanced Information Networking and Applications Workshops. – Bradford, 2009. – P. 1037 – 1042.
6. Случайно ли распределение гамма-всплесков по небу? / Л.Г. Балаж, Р. Ваврек, А. Межарос и др. // Астрофизический бюллетень. – 2010. – Т. 65. – №3. – С. 292–301.

7. Дорошенко А.Ю. Архітектура і операційні середовища комп'ютерних систем / А.Ю. Дорошенко, В.М. Кислоокий, О.Л. Синявський. – К.: Національний університет «Києво-Могилянська академія», 2005. – 220 с.

8. Букатов А.А. Программирование многопроцессорных вычислительных систем / А.А. Букатов, В.Н. Дацюк, А.И. Жегуло. – Ростов н/Д.: Издательство ООО «ЦВВР», 2003. – 208 с.

9. Столлингс В. Операционные системы / В. Столлингс. – [4-е изд.]. – М.: Издательский дом «Вильямс», 2002. – 848 с.

10. Многоуровневое структурное проектирование программ. Теоретические основы, инструментарий / Ющенко Е.Л., Цейтлин Г.Е., Грицай В.П., Терзян Т.К. – М.: Финансы и статистика, 1989. – 342 с.

11. Таненбаум Э. Архитектура компьютера / Э. Таненбаум. – [5-е изд.]. – СПб.: Питер, 2007. – 844 с.

12. MSDN: Parallel Programming in the .NET Framework [Електронний ресурс]. – Режим доступу: URL: http://msdn.microsoft.com/en-gb/library/Parallel Programming in the .NET Framework

13. Stephen Toub The Past, Present and Future of Parallelizing .NET Applications / Stephen Toub // MSDN Magazine. – Microsoft, 2011. – P. 72 – 77.

Надійшла до редакції 23.03.2012

С.Д. ПОГОРЕЛЫЙ, А.В. ПОТЕБНЯ
Киевский национальный университет имени Тараса Шевченко, г. Киев, Украина

Формирование и исследование параллельной схемы алгоритма Крускала для систем с общей памятью.

Выполнена формализация алгоритма Крускала построения минимального покрывающего дерева неориентированного графа с использованием математического аппарата модифицированных систем алгоритмических алгебр В.М. Глушкова. Предложена концепция его распараллеливания для архитектур компьютерных систем с общей памятью. Проведен ряд исследований алгоритма и сформированы его временные характеристики. Приведены рекомендации по использованию алгоритма при решении прикладных задач.

Ключевые слова: архитектуры с общей памятью, минимальное покрывающее дерево, алгоритм Крускала, поток, системы алгоритмических алгебр, РСА, ПРСА, эквивалентные преобразования схем, распараллеливание, граф

S.D. POGORILYI, A.V. POTEBNIA
Kyiv National Taras Shevchenko University, Ukraine

Formation and Investigation of Kruskal's Algorithm Parallel Scheme For Shared Memory Systems.

Formalization of Kruskal's algorithm for constructing a minimal spanning tree of an undirected graph using mathematical means of V.M. Glushkov's algorithmic algebras modified systems is done. The conception of its paralleling for shared memory computer systems architectures is proposed. A set of algorithm investigations is conducted and its temporal characteristics are formed. Recommendations of algorithm usage for applied tasks solving are provided.

Keywords: shared memory architectures, minimal spanning tree, Kruskal's algorithm, thread, modified systems of algorithmic algebras, RSA, PRSA, equivalent scheme transformation, paralleling, graph