УДК 618.3 518.6

ОДИН ПІДХІД РОЗВ'ЯЗАННЯ ЗАДАЧ ОБЧИСЛЮВАЛЬНОЇ ГЕОМЕТРІЇ НА ОСНОВІ РЕКУРСИВНО-ПАРАЛЕЛЬНОГО АЛГОРИТМУ

© В.М. Терещенко

Київський національний університет імені Тараса Шевченка
факультет кібернетики
Пр-т академіка Глушкова, 2, корпус 6, Київ, Україна
e-mail: *v_ ter@ukr.net*

Abstract

This article is devoted to the generalized approach of the effective decision of problems of computing geometry which initial data are set of points in Euclidean planes. The basis of this approach is construction recursion-parallel algorithm by means of strategy «distribute and dominate». In particular, on an example of a problem of a finding of a convex environment of set of points, it is offered recursion-parallel algorithm of its decision.

Введення

Постановка проблеми. Одним з важливих напрямків побудови ефективних алгоритмів розв'язання задач в області інформатики є створення паралельних алгоритмів. Особливістю таких задач є те, що при наявності їх внутрішнього паралелізму, вони мало досліджені щодо застосування паралельних алгоритмів їх розв'язання. Це важливо з тієї точки зору, що на сьогоднішній день обчислювальна геометрія має досить широку область застосувань.

Аналіз останніх досліджень та публікацій В ряді робіт [1] запропоновано ряд послідовних алгоритмів побудови опуклої оболонки на площині на основі стратегії «розподіляй та володарюй» з можливою перспективою їх розпаралелювання. При цьому, їх оцінка складності в найгіршому випадку складає $O(N \log N)$ з лінійним часом злиття. Зокрема в роботі Препарарати і Хонга [10] розглядається алгоритм побудови опуклої оболонки опуклих многокутників шляхом побудови опорних відрізків з лінійним часом злиття. Згідно аналізу побудови рекурсивно-паралельних алгоритмів вирішальним моментом ефективності являється саме час кроку злиття, а тому заслуговують уваги ідеї побудови алгоритмів які дозволяють одержати час кроку злиття менший за $O(N)$.

Метою даної статті є розгляд рекурсивно-паралельного алгоритму побудови опуклої оболонки для багатопроцесорної системи на основі стратегії «розподіляй та володарюй», крок злиття якого виконується за час $O(\log N)$. Така ефективність кроку злиття досягається за рахунок використання структури даних – зчіплювальна черга, яка дозволяє виконувати усі необхідні операції, в тому числі побудову опорних відрізків до двох оболонок, на кожному кроці за логарифмічний час. Це, в свою чергу, дає можливість розробити єдину схему побудови паралельних алгоритмів розв'язання задач обчислювальної геометрії для багатопроцесорних систем на основі стратегії «розподіляй та володарюй».

Постановка задачі. *На площині задана множина S із N точок. Розробити рекурсивно-паралельний алгоритм побудови опуклої оболонки заданої множини точок.*

1. Математична модель алгоритму

Математична модель алгоритму складається із трьох етапів: попередньої обробки, етапу рекурсивного розбиття множини точок та етапу злиття опуклих оболонок.

Попередня обробка

На етапі попередньої обробки формується структурований масив точок $U(\mathrm{P}_{ij})$ із упорядкованих по x та y координатах списків точок U_{x}, U_y, відповідно.

Нехай задана множина S із N точок $S = \{P_1, P_2, ..., P_N\}$ на площині (1). І нехай задано $O(N)$ процесорів: $\Pi_1, \Pi_2, ..., \Pi_{N/2}$. В результаті проведеної попередньої обробки, у запропонований спосіб, на вході алгоритму матимемо масив точок $U = \{P_{ij}, \quad i, j = 1, N\}$, де i – індекс, який указує номер точки у впорядкованому списку по x координаті, а j – індекс, який указує номер точки у впорядкованому списку по y координаті. На (1) показаний випадок для N=12.

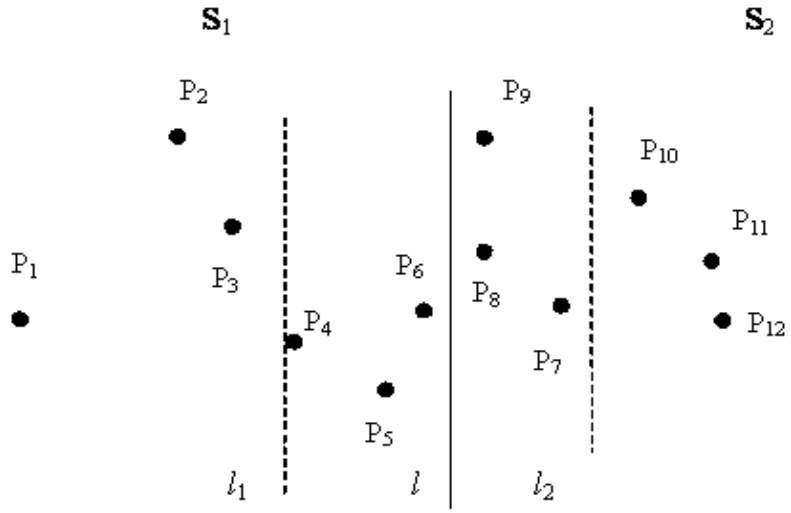

Рис. 1: Упорядкований список $U = \{P_{51}, P_{122}, P_{13}, P_{74}, P_{45}, P_{66}, P_{87}, P_{118}, P_{39}, P_{1010}, P_{211}, P_{912}\}$

Далі, на множині S із N даних будується бінарне дерево з коренем, (2), кожному вузлу якого відповідає ціле число k, відносно якого розбивається список точок у вузлах на два списки рівної потужності, відносно медіани, після порівняння перших індексів точок масиву P_{ij}. Число k визначається за один прохід по дереву, якщо відома кількість точок заданої множини, за формулою:

$$k = [(m + M)/2] \tag{1}$$

де m – номер першого елементу списку, M – останній елемент списку. Сформований у такий спосіб список подається на вхід алгоритму, граф якого має вигляд двійкового дерева, (2).

Розбиття множини точок (рекурсивний спуск)

На етапі розбиття на кожному кроці рекурсії задана множина точок, у вигляді списку U, розбивається на підмножини U_1, U_2 рівної потужності відносно медіани l, які передаються на наступний крок рекурсії. Процес розбиття завершується коли у підмножинах розбиття залишиться по одному елементу. Структура даних яка підтримує цей процес є наше дерево алгоритму, (2). У листках двійкового дерева будуть окремі точки з U. У випадку послідовного алгоритму загальний час рекурсивного спуску буде $O(N_2N)$, оскільки на кожному кроці рекурсії передача даних виконується за час $O(M)$, а пошук медіани на впорядкованому по координаті x індексованому масиві точок P_{ij} виконується за константний час $O(1)$. Медіана визначається по формулі

$$l = (Pkj + Pk + 1j)/2, \tag{2}$$

де k визнається по формулі (1), а M кількість точок у списку.

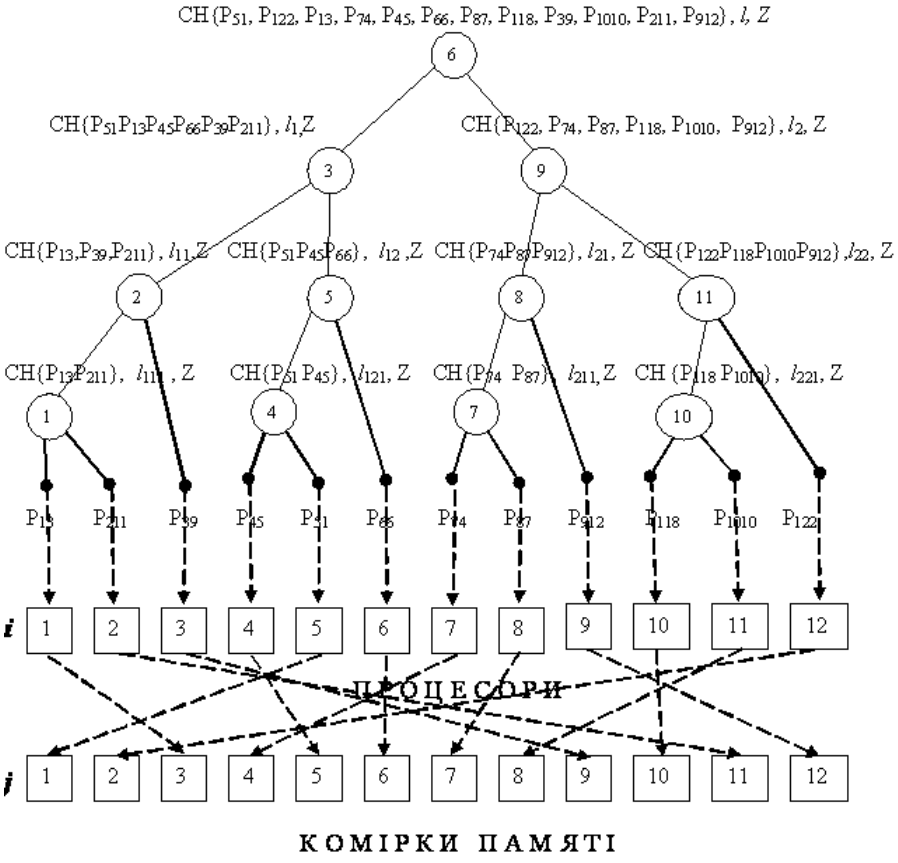

Рис. 2: Граф алгоритму

Час, необхідний на рекурсивний спуск у випадку паралельного алгоритму, визначається наступною лемою.

Лема 1. Етап рекурсивного розбиття множини S із N точок на підмножини S_1 та S_2 рівної потужності на площині, пошук медіани ℓ та передачу підмножин S_1 та S_2 за допомогою N процесорів можна виконати за час $O(\log_2 N)$, виконавши попередню обробку за час $O(N \log_2 N)$.

Доведення. Нехай задана множина точок у вигляді індексованого двовимірного, упорядкованого масиву $U = P_{ij}, \quad i, j = 1, N$, де i – індекс, який указує номер точки у впорядкованому списку Ux, а j – індекс, який указує номер точки у впорядкованому списку Uy. Таке представлення множини точок дозволяє, знаючи кількість точок N у списку U, побудувати дерево розбиття (двійкове дерево). На (2) побудоване дерево для N=12 точок.

Як показано на рисунку, з першим індексом i кожної точки P_{ij} пов'язаний номер процесора, а з другим індексом j позначений номер комірки пам'яті в яку заноситься точка згідно упорядкованості по y координаті. На кожному кроці розбиття відповідні процесори синхронно порівнюють перші індекси зі списку точок та розсилають точки у відповідні вузли алгоритму, зберігаючи при цьому порядок розташування точок, який визначається їх порядком в комірках пам'яті. Враховуючи чітку упорядкованість по обом індексам точок $_{Pij}$ масиву U та взаємозв'язок між процесорами та елементами пам'яті, час виконання процесу злиття у кожному вузлі дерева не перевищуватиме константу $O(1)$.

Таким чином, загальний час розбиття буде не більшим ніж $O(\log_2 N)$ для найгіршого вводу даних. Що і треба було довести.

□

Злиття результатів (рекурсивний підйом)

На кожному кроці рекурсивного підйому, починаючи з другого, на вхід батьківського вузла v дерева подаються опуклі оболонки від лівого сина $U_L(U_L = \text{ЛСИН}[v])$ та правого сина $U_R(U_R = \text{ПСИН}[v])$. Необхідно побудувати опуклу оболонку для вузла v.

Лема 2. Етап рекурсивного злиття результатів визначення опуклої оболонки множини S із N точок на площині за допомогою N процесорів, можна виконати за час $O(\log_2 N)$.

ПРОЦЕДУРА ЗЛИТТЯ ОПУКЛИХ ОБОЛОНОК. Для побудови опуклої оболонки опуклих оболонок синів вузла v, необхідно визначити верхні та нижні опорні вершини лівої та правої опуклих оболонок, розчепити взаємо опуклі ланцюги між цими опорними точками, і з'єднати відповідні ввігнуті частини лівої та правої опуклих оболонок верхнім та нижнім опорними відрізками. Для того щоб виконати такі дії необхідно: визначити структуру даних, яка б підтримувала б опуклу оболонку в кожному вузлі, дозволяла б знаходити опорні точки, розчіпляти та зчіпляти частини опуклих оболонок, проводити опорні відрізки.

При виборі структури даних, яка б виконувала вище наведені операції за логарифмічний час, скористаємось ідеєю, запропонованою в роботі Овермарса і Ван Лювена [11] для послідовного алгоритму, згідно якої за таку структуру даних обрано зчеплену чергу, із заданою на ній процедурою З'ЄДНАТИ (U_L, U_R).

В роботі доведено, що процедура З'ЄДНАТИ (U_L, U_R), починаючи з кореневих вершин збалансованих дерев, які підтримують U_L та U_R дозволяє побудувати верхню та нижню границі опуклої оболонки за час $O(\log N)$. При виконанні пошуку опорних вершин та побудові опорних відрізків, використовується класифікація вершин, подана в роботі [1] (3).

Необхідно лише до поданих дев'яти випадків класифікації при побудові верхньої опуклої оболонки додати 9 аналогічних випадків для побудови нижньої опуклої оболонки.

Після визначення опорних точок та розчепленням по ним опуклих оболонок синів вузла v, відкидається права частина дерев, які підтримують U_L між опорними точками та ліва частина дерев, які підтримують U_R між опорними точками. Збалансовані дерева, які підтримуватимуть верхню та нижню опуклу оболонку вузла v утворюються шляхом злиття частин дерев, які залишилися.

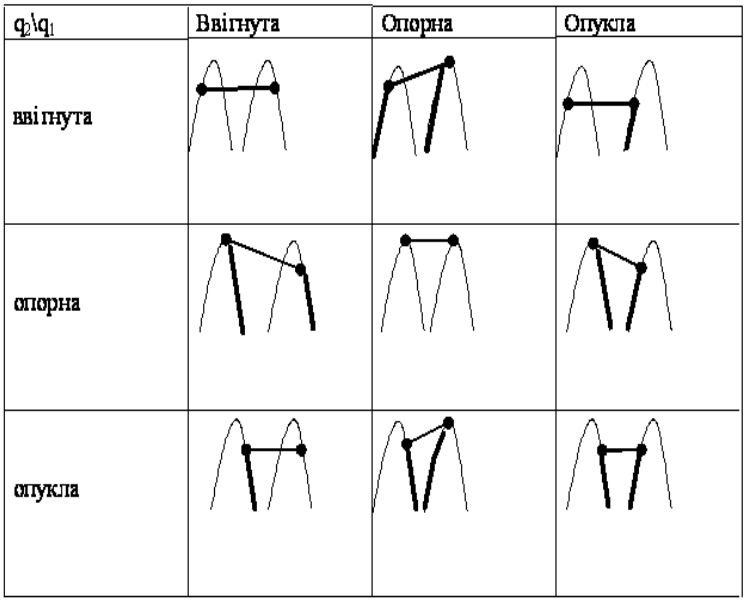

Рис. 3:

Знову ж таки, усі наведені операції виконуються за час $O(\log_2 N)$. Одержані у такий спосіб опуклі оболонки передаються на наступний рівень рекурсії.

На рисунку (4) показано крок злиття алгоритму. У кожному вузлі позначено функцію З'ЄДНАТИ (U_L, U_R) через Z, а відповідні границі опуклих оболонок через $CH(U)$; окрім, цього через l_{ij} позначено медіани. На (5) показано пошук опорних вершин лівої та правої опуклих оболонок (4) на кроці злиття. По вершинам $\mathrm{P}_2, \mathrm{P}_5$

розчіпляється ліва опукла оболонка $U_L = CH(S_1)$ на ланцюги $U_{L1} = (\mathrm{P}_2, \mathrm{P}_1, \mathrm{P}_5)$ та $U_{L2} = (\mathrm{P}_5, \mathrm{P}_6, \mathrm{P}_2)$, а права опукла оболонка $U_R = CH(S_2)$ розчіпляється по вершинам $\mathrm{P}_9, \mathrm{P}_{12}$ на ланцюги $U_{R1} = (\mathrm{P}_9, \mathrm{P}_8, \mathrm{P}_7, P_{12})$ та $U_{R2} = (\mathrm{P}_{12}, \mathrm{P}_{11}, \mathrm{P}_{10}, P_9)$. Ланцюги $U_{L2} = (\mathrm{P}_5, \mathrm{P}_6, \mathrm{P}_2)$ і $U_{R1} = (\mathrm{P}_9, \mathrm{P}_8, \mathrm{P}_7, P_{12})$ взаємо опуклі то згідно алгоритму вони відкидаються, а значить, і зникають відповідні вузли дерев які підтримують відповідні опуклі оболонки.

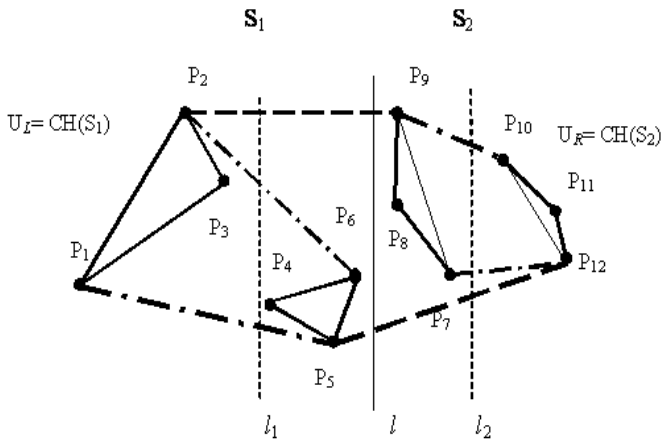

Рис. 4:

Так у дереві, яке підтримує ліву опуклу оболонку зникнуть вузли, які знаходяться праворуч опорних точок P_2, P_5 – це вузол P_6, а у дереві правої опуклої оболонки зникнуть вузли, точки яких розташовані ліворуч від правих опорних точок – це вузли P_8, P_7. Після цього зчіплюються ланцюги $U_{L1} = (\mathrm{P}_2, \mathrm{P}_1, \mathrm{P}_5)$ і $U_{R2} = (\mathrm{P}_{12}, \mathrm{P}_{11}, \mathrm{P}_{10}, P_9)$ за допомогою опорних відрізків (P_2, P_9) та (P_5, P_{12}), які і утворять результуючу границю опуклої оболонки $CH(S) = (\mathrm{P}_{12}, \mathrm{P}_{11}, \mathrm{P}_{10}, P_9, \mathrm{P}_2, \mathrm{P}_1, \mathrm{P}_5)$.

Частини дерев що залишились зіллються в єдине збалансоване дерево, яке підтримує одержану границю кінцевої опуклої оболонки $CH(S)$.(6).

Процеси, які реалізують алгоритм на *етапі злиття* можна описати таким чином:

Теорема 1. *Задачу побудови опуклої оболонки на множині S із N точок площини можна розв'язати за допомогою N процесорів із часом $O(\log_2 N)$, із використанням $O(N)$ пам'яті за два рекурсивні проходи, виконавши попередньо обробку вхідних даних за час $O(N \log_2 N)$.*

У випадку паралельної обробки, синхронно може працювати менша кількість процесорів, ніж для вузлових списків, в одних і тих же вузлах алгоритму, а завантаженість процесорів буде більш висока. Питання використання більш оптимального складу процесорів (комп'ютерів) у даній роботі не розглядалось й потребує подальших досліджень. В той же час відомо з робіт [3,4], що якщо позначити за k кількість процесорів, за w – загальну кількість операцій алгоритму, а за $O(F(N))$ – складність алгоритму для необмеженої кількості процесорів, то час виконання алгоритму

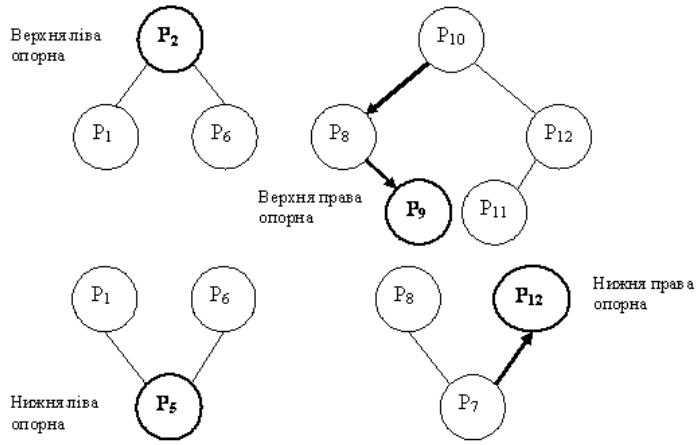

Рис. 5:

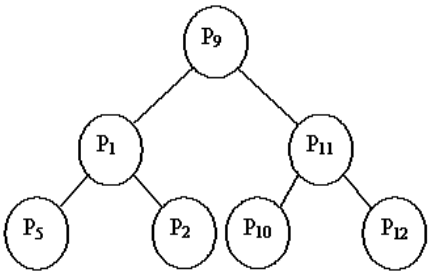

Рис. 6:

можна представити у вигляді:

$$O(f(N)) = ((k-1)O(F(N)) + w)/k \tag{3}$$

У нашому випадку, згідно теореми 2, $F(N) = \log_2 N$, а значить час виконання алгоритму для k процесорів визначатиметься наступним співвідношенням:

$$O(f(N)) = ((k-1)O(\log_2 N) + w)/k \tag{4}$$

Так, зокрема, для кількості операцій w порядку $O(N)$ із співвідношення (4) випливає: у випадку $k = 2$ процесорам, час виконання алгоритму буде рівний $O(N/2)$, а при $k = N/2 - O(\log_2 N)$. Окрім цього, на наш погляд, синхронна обробка даних алгоритму може бути замінена конвеєрною, що розширює можливості підвищення загальної ефективності алгоритму.

2. Реалізація алгоритму

Що до практичної реалізації розробленого паралельного алгоритму, то одним з ефективних підходів може бути підхід на основі використання технології ПАРКС (Паралельні Асинхронні Рекурсивно Керовані Системи) [5], яка дозволяє

досить ефективно реалізувати саме рекурсивно-паралельні алгоритми розв'язання складних задач. Про це свідчать відомі результати при розв'язанні задач обчислювальної математики [5] та оптимізаційних задач в економіці [6]. На сьогоднішній день, час новітніх технологій, ідея ПАРКС набула свого нового дихання у системах ПАРКС-JAVA [7, 8], що дає можливість розпаралелювати звичайні рекурсивні алгоритми для паралельного виконання на комп'ютерах, з'єднаних через локальну мережу, або мережу Internet.

Система ПАРКС-JAVA реалізується на основі локальної чи глобальної комп'ютерної мережі обробки інформації, надаючи можливість користувачам малопотужних комп'ютерів використовувати ресурси мультипроцесорних комплексів, або ресурси комп'ютерної мережі. Основні елементи системи ПАРКС та засоби паралельного програмування для реалізації ПАРКС-технології, детально описані в роботах [5-8].

Висновки

В даній роботі запропонована ідея розробки узагальненого алгоритму розв'язання задач обчислювальної геометрії на прикладі задачі про побудову опуклої оболонки на основі схеми «розподіляй та володарюй» при використанні технології ПАРКС. Зважаючи на те, що більшість задач обчислювальної геометрії пов'язані з обробкою множин точок, запропонована стратегія може бути загальною методологією розв'язання широкого класу задач обчислювальної геометрії.

Так в роботі, завдяки побудові структурованого масиву точок заданої множини та вдало вибраних структур даних: дерева алгоритму та зчепленої черги, яка описує опуклу оболонку у кожному вузлі дерева алгоритму, вдалось одержати оптимальний час кроку злиття $\theta(\log_2 N)$ для багатопроцесорної системи.

Етап розбиття, запропонований у цій роботі, може бути загальним як для задач обчислювальної геометрії, вхідними даними яких є точки, так і може бути узагальненим і на розпаралелювання задач інших класів [9]. Основні кроки етапу злиття теж можуть бути спільними для багатьох класів задач обчислювальної геометрії. Відмінними лише будуть процедури злиття, які можна розробляти, а потім використовувати як окремі модулі програми.

Список литературы

1. *Препарата Ф., Шеймос М.* Вычислительная геометрия: Введение. М.: Мир, 1989. – 478 с.
2. *Анісімов А.В., Терещенко В.М., Кравченко І.В.* Основні алгоритми обчислювальної геометрії. ВПЦ "Київський університет", Київ, 2002, 81 с.
3. *В. В. Воеводин, Вл. В. Воеводин.* Параллельные вычисления, СПб., БХВ – Петербург, 2004, 608 с.
4. *В. В. Воеводин.* Матричные методы и алгоритмы. Сб. научных трудов // РАН Инс. Выч. Мат., под ред. Воеводина, 1993, 171 с.
5. *Анисимов А.В., Глушков В.М.* Управляющие пространства в асинхронных параллельных вычислениях. // Кибернетика, №5, 1980.-С.1-9.

6. *Анисимов А.В., Борейша Ю.Е., Карапетян М.С.* Параллельное программирование экономических систем на основе декомпозиционных методов. КИБЕРНЕТИКА, Киев, Наукова думка, 1990, №3, с. 105-110.
7. *Анисимов А.В., Деревянченко А.В.* Построение виртуального параллельного пространства с использованием технологии ПАРУС-JAVA. // Материалы Международной научно-технической конференции ИМС, 2003,Т.2.-С 18-19.
8. *Анісімов А.В., Дерев'янченко А.В.* Система ПАРКС-JAVA як засіб вирішення паралельних алгоритмів на комп'ютерній мережі. // Материалы четвертой международной научно-практической конференции УкрПРОГ'2004, ПРОБЛЕМЫ ПРОГРАММИРОВАНИЯ, №2-3, 2004.-С.282-284.
9. *Анісімов А.В., Кулябко П.П., Терещенко В.М.* Паралельні алгоритми в дослідженнях неперервних систем. К.: Видавництво "ЛОГОС", 1999.-50с.
10. *F. P. Preparata fnd J. Hong.* Convex hulls of finste sets of points two and three dimensions, Comm. ACM 2(20),87 -93(Feb. 1977).
11. *M. H. Overmars and J. van Leeuwen.* Maintenance of configurations in the plane, J.Cjmput. and Syst. Sci. 23, 166-204(1981).

Статья поступила в редакцию 17.12.2007