Науково-практичний звіт на тему

Найбільший порожній прямокутник (АО6)

Д. О. Чучман, Студент 3 курсу, групи КМ1

Анотація. У роботі проводиться аналіз та порівняння основних алгоритмічних підходів до розв’язання оптимізаційної задачі «Найбільший порожній прямокутник» (РМП (АО6)) . Детально розглядаються два ключові методи: алгоритм Орловського, що базується на переборі «дотичних прямокутників»; метод на основі Діаграми Вороного, що оптимізує пошук через аналіз суміжних комірок. Для кожного методу наводиться опис основної ідеї та оцінка обчислювальної складності. Реалізовано графічний інтерфейс, що забезпечує наочну візуалізацію результатів і збереження даних у файл.

Abstract. This paper provides an analysis and comparison of the main algorithmic approaches to solving the "Largest Empty Rectangle" (LER (AO6)) optimization problem. Two key methods are examined in detail: Orlowski's algorithm, which is based on the enumeration of "tangent rectangles," and a method based on the Voronoi Diagram, which optimizes the search through the analysis of adjacent cells. For each method, a description of the main idea and an assessment of its computational complexity are provided. A graphical interface has been implemented, which provides visual visualization of results and saving data to a file.

1. Вступ

Постановка проблеми. В роботі розглядається задача знаходження найбільшого порожнього прямокутника (РМП (АО6)). Формально вона полягає в тому, щоб на заданій обмеженій (прямокутником чи опуклою оболонкою) множині N із *n* точок на площині побудувати прямокутник найбільшої площі, який би не містив всередині жодної точки множини N.

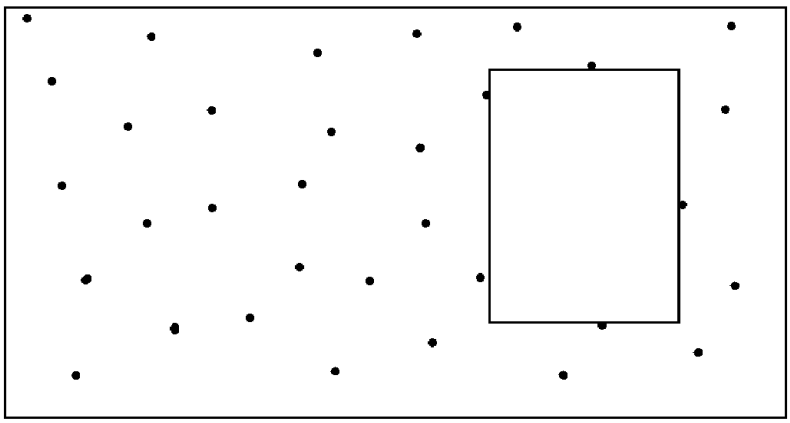

Рис 1.

Розв'язки цієї задачі мають велике теоретичне і практичне значення. Вони часто виникають, коли необхідно знайти оптимальне розташування для нового об'єкта, наприклад, магазину чи стадіону, або при створенні нового виробництва, якщо бажано уникнути конкуренції за вільний простір. Також задача застосовується для пошуку місця розташування шкідливих підприємств, які мають бути якнайдалі від населених пунктів. Окрім цього, задача знаходження найбільшого порожнього прямокутника виникає при роботі з базами даних.

Аналіз останніх досліджень. Незважаючи на значну практичну важливість задачі, пошук ефективних алгоритмів для **загального випадку** (де прямокутник

може бути довільно орієнтований) виявився не простим завданням. Варто зазначити, що для простіших, обмежених версій цієї задачі були знайдені ефективні, а іноді й оптимальні розв'язки. Наприклад, для пошуку найбільшого вписаного опуклого многокутника або найбільшого ортогонального «сходинкового» многокутника існують алгоритми з лінійною часовою складністю $O(n)$.

Для загальної задачі РМП (АО6), яка розглядається в цій роботі, на сьогоднішній день можна виділити декілька основних перспективних шляхів для її вирішення.

1) **Метод «Розділяй та володарюй»:**

Це один з основних підходів до задачі. Одними з перших цю задачу дослідили Chazelle, Drysdale та Lee які у 1986 [1] році запропонували алгоритм на основі цього методу з часовою складністю $O(n\log^3 n)$(Рис. 2). Пізніше Aggarwal та Suri [2] змогли покращити цю теоретичну оцінку, досягнувши загальної часової складності $O(n\log^2 n)$. Однак, як зазначали самі автори, їхній алгоритм має дуже велику приховану константу, тому для практичної реалізації попередній алгоритм з оцінкою $O(n\log^3 n)$ може виявитися ефективнішим.

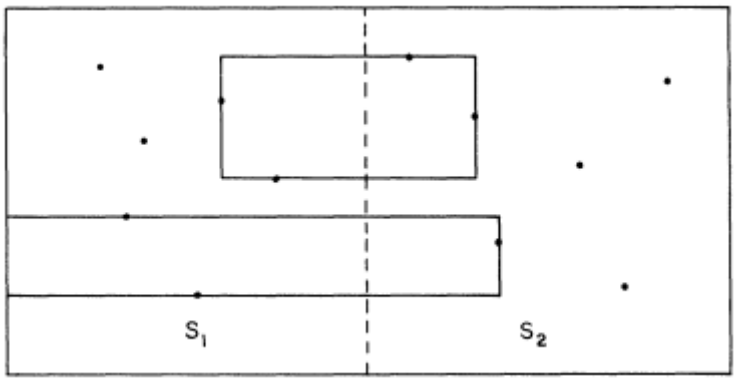

Рис. 2

2) **Метод «Перебору всіх дотичних прямокутників»:**

Другий основний підхід, вперше запропонований Орловським [3]. Ідея полягає в тому, що розв'язок задачі гарантовано буде «дотичним» — тобто кожною своєю стороною торкатиметься точки з вхідної множини або межі області . Алгоритм Орловського, який використовує спеціалізовану структуру даних «висотне дерево», має часову складність $O(n^2)$ у найгіршому випадку та O$(n\log n)$ у середньому .

3) **Інші підходи:**

Окрім цих двох основних напрямків, існують і альтернативні методи. Наприклад, підхід на основі Діаграми Вороного, який зводить задачу до аналізу суміжних комірок Вороного і має складність $O(n^2\log n)$ [4]. Також розглядається метод з використанням триангуляції , який дозволяє досягти складності $O(n\log n)$ [5].

Рис. 3

Мета статті. Розробити та проаналізувати ефективні алгоритмічні підходи до розв'язання задачі пошуку найбільшого порожнього прямокутника (РМП (АО6)) на заданій множині точок. Основна увага приділяється опису та порівнянню двох методів: алгоритму Орловського та алгоритму на основі Діаграми Вороного, а також оцінці їх обчислювальної складності.

2. Основна частина

Постановка Задачі. На заданій обмеженій (прямокутником чи опуклою оболонкою) множині N із *n* точок на площині побудувати прямокутник найбільшої площі, який би не містив всередині жодної точки множини N.

2.1. Алгоритм Орловського (метод перебору дотичних прямокутників)

Для розв’язання задачі про РМП можна скористатись методом «перебору всіх дотичних прямокутників», який був вперше запропонований Орловським [3] . Для цього введемо ключове поняття дотичного прямокутника.

Означення 1. Дотичним прямокутником називається прямокутник, який кожною стороною дотикається до точки (з множини N) або до сторони оточуючого прямокутника, проте не повністю міститься в оточуючому прямокутнику і не містить жодної точки всередині.

В основі методу лежать дві ключові леми:

Лема 1. Розв’язком задачі про РМП гарантовано буде дотичний прямокутник.

- **Доведення:** Припустимо, від супротивного: ми маємо розв’язок, який не є дотичним прямокутником. Це означає, що існує хоча б одна сторона, в напрямку якої він може бути розширений, не захопивши нових точок. Але це суперечить умові, що наш прямокутник вже має максимальну площу. Отже, ми прийшли до протиріччя.

Лема 2. Для *n* точок існує щонайбільше n^2 дотичних прямокутників.

- **Доведення:** Можна побачити, що дотичний прямокутник повністю визначається своїми протилежними сторонами. Кожна сторона, у свою чергу, фіксується або точкою з множини N, або стороною оточуючого прямокутника. Отже, кількість таких комбінацій не може перевищувати $n \cdot n$ тобто n^2.

З цих лем випливає, що для вирішення задачі достатньо перебрати всі можливі дотичні прямокутники. Орловський вперше показав [3], як це зробити за допомогою двох відсортованих списків (по *x* та *y*). У цьому ж посібнику пропонується інший підхід — використання структури даних «висотне дерево».

Означення 2. Висотним деревом називається геометрична структура даних (бінарне дерево), яка зберігає точки на площині та має наступні властивості:

1. Це бінарне дерево.
2. Лівий син знаходиться нижче (менша *y*-координата) та лівіше (менша *x*-координата).
3. Правий син знаходиться не вище (*y*-координата не більша) та не лівіше (*x*-координата не менше).

Лема 3. Для будь-якого набору точок висотне дерево може бути побудовано за час $O(n \log n)$.

- **Доведення:** Для побудови дерева достатньо виконати два кроки:
- 1. Відсортувати точки по парі (*x*, *y*).
- 2. Будувати висотне дерево, на кожному кроці додаючи по одній точці з відсортованого списку (що займає час $O(n \log n)$ на одну точку) до дерева.

Для опису роботи алгоритму також потрібні наступні означення:

Означення 3. «Ліва вправо» гілка — це список вершин, який складається з лівого сина, правого сина лівого сина, правого сина правого сина лівого сина і так далі аж до кінця.

Означення 4. «Права вліво» гілка — це список вершин, який складається з правого сина, лівого сина правого сина, лівого сина лівого сина правого сина і так далі.

На цих гілках базується ключова властивість дерева:

Лема 4. Всі дотичні прямокутники, які верхньою стороною дотикаються до кореня висотного дерева, іншими сторонами дотикаються або до сторін оточуючого прямокутника, або до точок гілок «ліва вправо» та «права вліво».

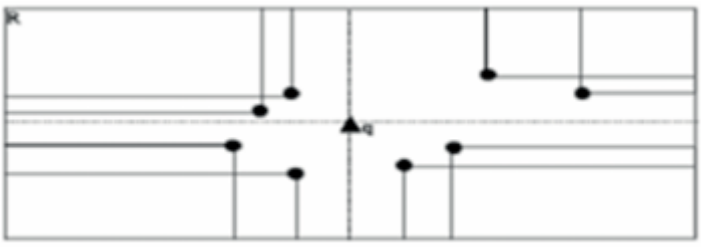

Рис. 3

- **Доведення:** Звернувши увагу на означення гілок, можна побачити, що між лівою та правою гілками гарантовано відсутні точки. Оскільки дотичні прямокутники не містять жодної точки всередині, то всі вони будуть формуватись лише точками цих гілок та сторонами оточуючого прямокутника.

Корінь висотного дерева може бути видалений за лінійний час відносно суми точок у гілках («зшивання віток»). Отже, ми можемо розглянути всі дотичні прямокутники, що зверху дотикаються до кореня, потім видалити корінь та повторити процес. В результаті, ми розглянемо всі дотичні прямокутники, окрім тих, що верхньою стороною спираються на верхню сторону оточуючого прямокутника (їх ми розглядаємо окремо).

Алгоритм Орловського (покроково)

1. Розглянути дотичні прямокутники, що дотикаються верхньою стороною до верхньої сторони оточуючого прямокутника.
2. Побудувати висотне дерево для множини N. Ця структура даних дозволить ефективно впорядкувати точки.
3. Перевірити дотичні прямокутники, що дотикаються верхньою стороною до кореня висотного дерева. Це ядро методу, де ми використовуємо властивості гілок дерева.
4. Видалити корінь висотного дерева.
5. Якщо дерево містить хоча б одну точку, рекурсивно перейти до пункту 3.

Оцінка складності алгоритму

Теорема 1. Пошук найбільшого порожнього прямокутника на множині N із n точок на площині можна виконати алгоритмом Орловського за час $O(n^2)$ у найгіршому та $O(n \log n)$ у середньому випадках.

Доведення:

- Пункт 1) алгоритму можна виконати за $O(n)$ (за умови, що точки відсортовані по x).
- Пункт 2) (побудова дерева) виконується за $O(n \log n)$.
- Інші пункти алгоритму (3-5) перебирають дотичні прямокутники, витрачаючи на кожний константну кількість часу.
- Отже, загальний час роботи рівний $O(n \log n + s)$ де s — кількість дотичних прямокутників. Як було показано у Леми 2, у найгіршому випадку s може сягати n^2. Звідси оцінка складності у найгіршому випадку буде $O(n^2)$ та $O(n \log n)$ в середньому. Алгоритм також потребує $O(n)$ пам'яті.

ВИСНОВКИ ПО МЕТОДУ 1:

Описаний алгоритм дозволяє розв'язати задачу пошуку прямокутника максимальної площі. Часова оцінка алгоритму в гіршому випадку ($O(n^2)$)

використовуючи метод «перебору всіх дотичних прямокутників», не може бути покращена, оскільки на розгляд одного дотичного прямокутника витрачається константна кількість часу.

2.2. Алгоритм на основі Діаграми Вороного

В основі цього підходу лежить властивість, яка спочатку розглядається для простого випадку з чотирма точками, а потім оптимізується для *n* точок за допомогою Діаграми Вороного.

Основна ідея та випадок для 4-х точок

Розглянемо спочатку задачу для випадку, коли множина N складається лише із чотирьох точок. Для існування розв'язку необхідно, щоб ці чотири точки утворювали опуклий чотирикутник.

- **Твердження 1:** Можна довести, що для прямокутника максимальної площі, сторони якого проходять через вершини даного опуклого чотирикутника, виконується наступна властивість: одна з пар протилежних сторін прямокутника **перпендикулярна** до діагоналі опуклого чотирикутника.
- Іншими словами, як було показано у попередніх дослідженнях [9], одна пара сторін прямокутника буде **паралельна** одній з діагоналей опуклого многокутника, утвореного з цих точок.

Наївний підхід (складність O(n^3))

Спираючись на цю властивість, можна побудувати наївний алгоритм для *n* точок, перевіривши дію цього твердження для всіх пар:

1. Візьмемо довільні дві точки *p1* та *p2* і розглянемо відрізок *p1p2*. Ми можемо вважати його діагоналлю шуканого чотирикутника.
2. Побудуємо прямі *l1* та *l2* що проходять через *p1* та *p2* і є перпендикулярними до прямої *p1p2*
3. Знайдемо всередині смуги (між *l1* та *l2* точки *p3* та *p4* які є найближчими до прямої *p1p2* з різних сторін.
4. Проведемо через *p3* та *p4* прямі *l3* та *l4* які перпендикулярні до *l1* та *l2*.
5. Чотири прямі *l1*, *l2*, *l3*, *l4* визначають прямокутник, який є можливим розв'язком задачі.

Недоліки цього підходу:

- Точки *p3* та *p4* не завжди існують.
- Якщо вони не існують, пара точок *p1* та *p2* не може утворювати діагональ, що перпендикулярна до сторін розв'язку.
- Необхідно перебрати **всі пари** точок ($O(n^2)$) і для кожної пари знайти кандидата ($O(n)$), що дає загальну складність ($O(n^3)$).

Оптимізація за допомогою Діаграми Вороного

Оскільки підхід O(n^3) є занадто повільним, необхідно оптимізувати перебір пар точок. Тут на допомогу приходить Діаграма Вороного [4].

1. Припустимо, ми побудували для нашої множини діаграму Вороного.
2. Розглянемо точки *p1* та *p2*та відповідні їм многокутники Вороного *V1* та *V2*.
3. **Ключова властивість:** Можна показати, що точки *p3* та *p4* (найближчі до *p1p2* з різних сторін), які ми шукали у наївному підході, будуть точками, що відповідають многокутникам Вороного *V3* та *V4*, які є **суміжними одночасно до *V_1* та *V_2*** [4].
4. **Оптимізація (Чазелле):** Більш того, було показано (зокрема, у роботах Чазелле [6]), що якщо ми вже знаємо одну з точок (наприклад *p1*), через яку проходить ребро максимального порожнього прямокутника, то три інші точки повинні бути серед точок, що відповідають тим многокутникам Вороного, які є **суміжними до *V1*** (1-го порядку), а також тим, які є **суміжними до цих, перших суміжних** многокутників (2-го порядку).

Це дозволяє значно скоротити кількість пар точок, що перевіряються.

Алгоритм на основі Діаграми Вороного (покроково)

1. **Крок 1: Побудова Діаграми Вороного.** ЇЇ можна представити реберним списком з подвійними зв‘язками (РСПЗ).
2. **Крок 2: Ітеративний пошук кандидатів.** Перебираємо всі многокутники Вороного. Для точки *p_i*, що відповідає кожному такому многокутнику, знаходимо інші три точки-кандидати. Для цього нам треба лише перебрати всі суміжні (до другого порядку) многокутники, користуючись процедурою, описаною в "наївному підході".
3. **Крок 3: Перевірка та оновлення максимуму.** Подібно до того, як ми вже згадували, такі три точки не завжди існують.
    - Якщо вони не існують, ми просто переходимо до наступного многокутника Вороного.
    - Інакше, ми порівнюємо площу новознайденого прямокутника (*S*) з попередньою максимальною площею (s_{max}). Якщо *S* > s_{max} то ми запам‘ятовуємо новознайдений прямокутник, як максимальний (s_{max}= *S*).
4. **Крок 4: Пошук по парах суміжних точок.** Перебираємо всі пари суміжних точок і, аналогічно, пари точок, які суміжні одночасно з двома цими точками. Якщо таких (одночасно сумісних точок) немає, просто переходимо до наступної пари.
5. **Крок 5: Побудова прямокутника.** Знаходимо прямокутник, вписаний у ці 4 точки, сторона якого паралельна одній із діагоналей многокутника, описаного навколо цих чотирьох точок.

6. **Крок 6: Оновлення максимуму.** Порівнюємо площу S цього прямокутника із s_{max}, отриманою на попередніх кроках. Якщо $S > s_{max}$, то покладаємо $s_{max} = S$ і запам'ятовуємо цей прямокутник.
7. **Крок 7: Повторення.** Беремо наступну пару суміжних точок і повторюємо з кроку 4, поки всі пари не будуть перевірені.

Оцінка складності алгоритму

Теорема 2. Пошук найбільшого порожнього прямокутника на множині N із n точок на площині можна виконати алгоритмом на основі діаграми Вороного за час $O(n^2 \log n)$ у найгіршому випадку.

- **Доведення:** Діаграму Вороного (Крок 1) можна побудувати за час $O(n \log n)$. Оскільки кожен многокутник Вороного може містити не більше ніж $O(n)$ ребер (у середньому), то для перегляду кожної групи точок-кандидатів (Кроки 2-7) необхідний лінійний час $O(n)$. Оскільки ми робимо це для $O(n)$ точок, звідси отримуємо загальну складність алгоритму O $(n^2 \log n)$.

Висновки по методу 2:

Представлений алгоритм розв'язує задачу знаходження найбільшого порожнього прямокутника для скінченної множини точок на площині. Проте, слід зазначити, що більш популярною (і корисною) є задача про максимальний порожній прямокутник в дещо іншій постановці: на площині задана множина N, що міститься всередині деякого прямокутника R, і потрібно знайти прямокутник максимальної площі зІ сторонами, паралельними сторонам R.

Для цієї, іншої (і простішої), задачі існують швидші алгоритми зі складністю $O(n \log n)$, як було доведено, наприклад, Aggarwal та Suri [2]. Наш, більш загальний, варіант задачі (де прямокутник може бути повернутий довільно) є класичним і також розглядався у багатьох наукових роботах. Також слід зауважити, що описаний алгоритм чудово ілюструє одне з багатьох потужних застосувань діаграми Вороного в обчислювальній геометрії.

2.3. Алгоритм з використанням тріангуляції

Третій підхід до розв'язання задачі РМП базується на розбитті вхідної множини точок на простіші геометричні фігури. Замість того, щоб аналізувати всю множину одразу, цей метод використовує тріангуляцію як проміжну структуру для ефективного пошуку найбільшого порожнього простору.

Алгоритм (покроково):

Процес можна розділити на п'ять основних етапів:

1. **Побудова межі:** Спочатку будується опукла оболонка (наприклад, методом Грехема [10]), щоб визначити зовнішні межі множини точок N.

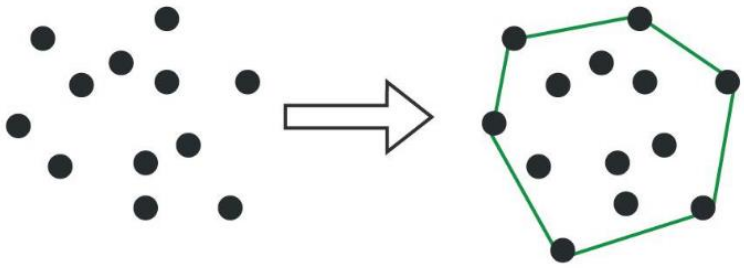

Рис. 4

2. **Розбиття (Триангуляція):** Далі, вся множина точок, що знаходиться всередині цієї оболонки, триангулюється. Це розбиває весь внутрішній простір на множину суміжних трикутників.

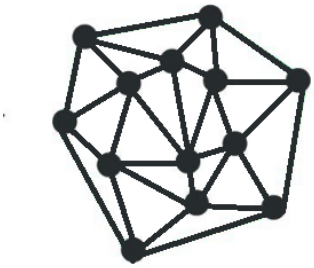

Рис. 5

3. **Пошук «зерна»:** Виконується аналіз отриманої тріангуляції з метою знайти трикутник з найбільшою площею . Цей трикутник слугуватиме основою для майбутнього прямокутника.

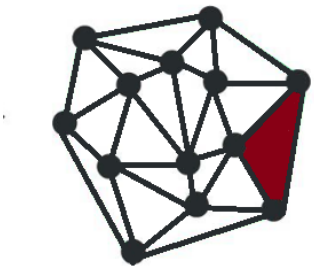

Рис. 6

4. **Розширення області:** Знайдений трикутник-кандидат "розширюється" у всіх напрямках, приєднуючи сусідні порожні трикутники, доки не утворить максимальну порожню область, яка його містить, але не включає жодної точки з N.

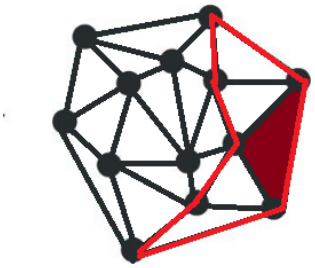

Рис. 7

5. **Фінальне вписання:** Вже в межах цієї нової, часто складної за формою, порожньої області виконується вписання прямокутника найбільшої можливої площі за допомогою оптимізаційної функції.

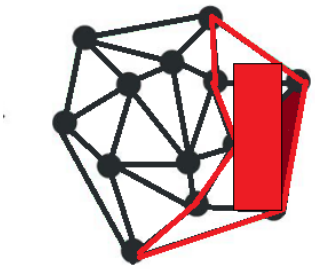

Рис. 8

Теорема 3. Загальна часова складність цього методу оцінюється як $O(n \log n)$ у найгіршому випадку .

- **Доведення:** Ця оцінка базується на тому, що найскладніші етапи — це початкова підготовка даних.

1. Побудова опуклої оболонки.
2. Подальша триангуляція множини точок виконуються за час $O(n \log n)$.
3. Подальші кроки, такі як пошук найбільшого трикутника, його розширення та фінальне вписання прямокутника (Кроки 3-5), можуть бути виконані ефективніше, за лінійний час $O(n)$. Таким чином, загальна складність алгоритму визначається — кроками 1 та 2.

3. Практична частина

3.1. Алгоритм роботи програми

Програма реалізує демонстрацію розв'язання задачі «Найбільший порожній прямокутник»

Після запуску програми користувач бачить головне вікно, розділене на дві частини: ліворуч — «Панель керування», праворуч — «Робоча область» (полотно для малювання).

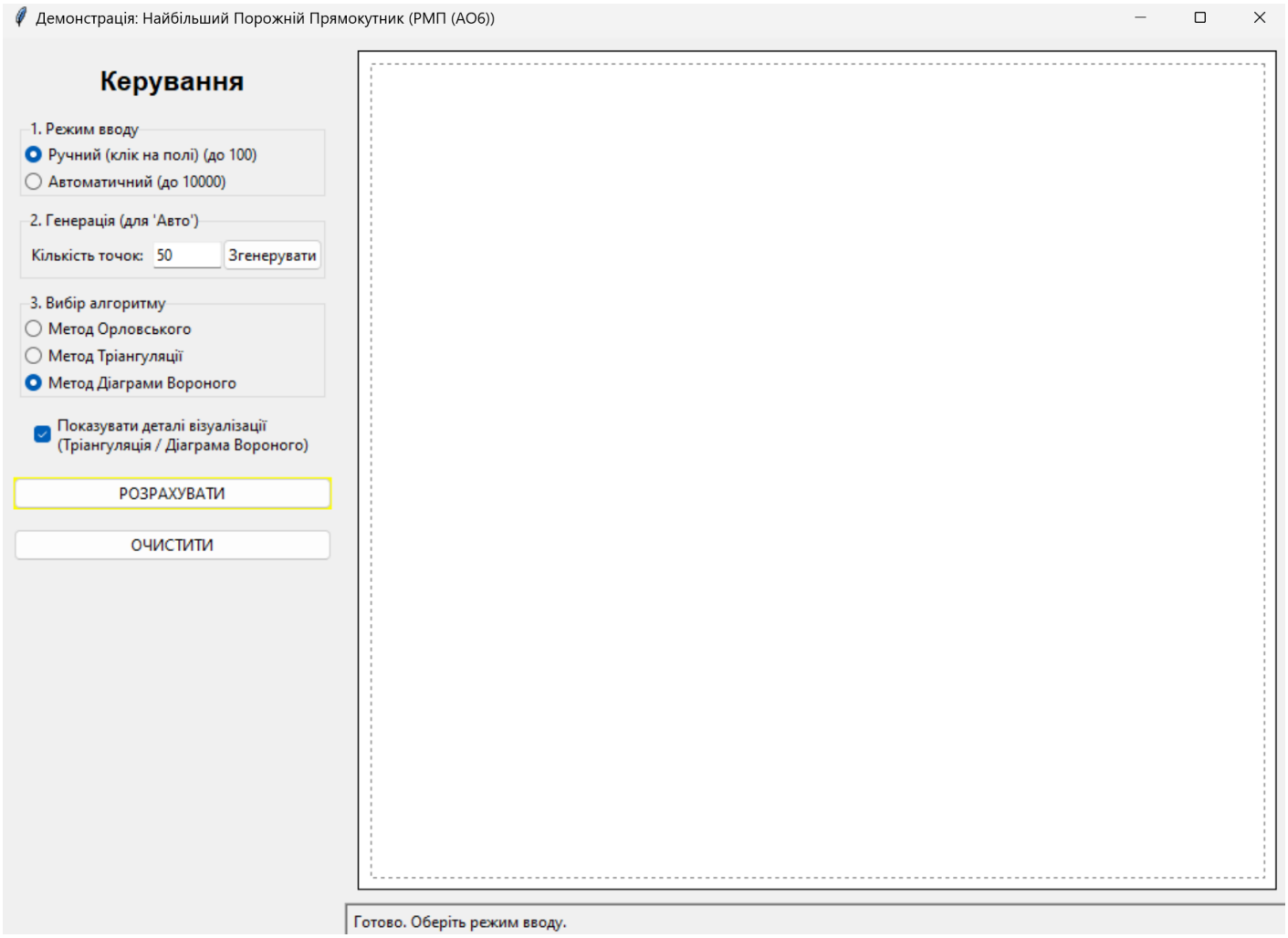

Рис.9

Робота з програмою складається з наступних кроків:

1) Ввід даних : Користувач обирає один з двох режимів вводу:

A. **«Ручний (клік на полі)»:** Дозволяє вводити до 100 точок, клікаючи мишею всередині пунктирної рамки робочої області.

B. **«Автоматичний (до 10000)»:** Дозволяє згенерувати велику кількість даних для тестування ефективності.

2) Генерація: Якщо обрано автоматичний режим, користувач вводить бажану кількість точок у поле «Кількість точок» (наприклад, 50, 5000 або 10000) і натискає кнопку «Згенерувати». Програма очищує полотно і створює вказану кількість випадкових точок у межах робочої області.

3) Вибір алгоритму: Користувач обирає один з трьох демонстраційних алгоритмів, для порівняння їх ефективності:

- Метод Орловського.
- Метод Тріангуляції.

- Метод Діаграми Вороного.

4) Вибір візуалізації: Користувач може увімкнути/вимкнути галочку «Показувати деталі візуалізації». Якщо опція увімкнена, програма *додатково* намалює Діаграму Вороного або Тріангуляцію.

5) Розрахунок: Користувач натискає кнопку «РОЗРАХУВАТИ». Програма засікає час, виконує обраний алгоритм і малює результат.

6) Результат: На полотні з'являється червоний прямокутник, що позначає знайдений найбільший порожній прямокутник. Якщо деталі були увімкнені, також з'являється зелена сітка Діаграми Вороного або блакитна сітка Тріангуляції. У статус-барі внизу вікна відображається час виконання алгоритму та площа знайденого прямокутника.

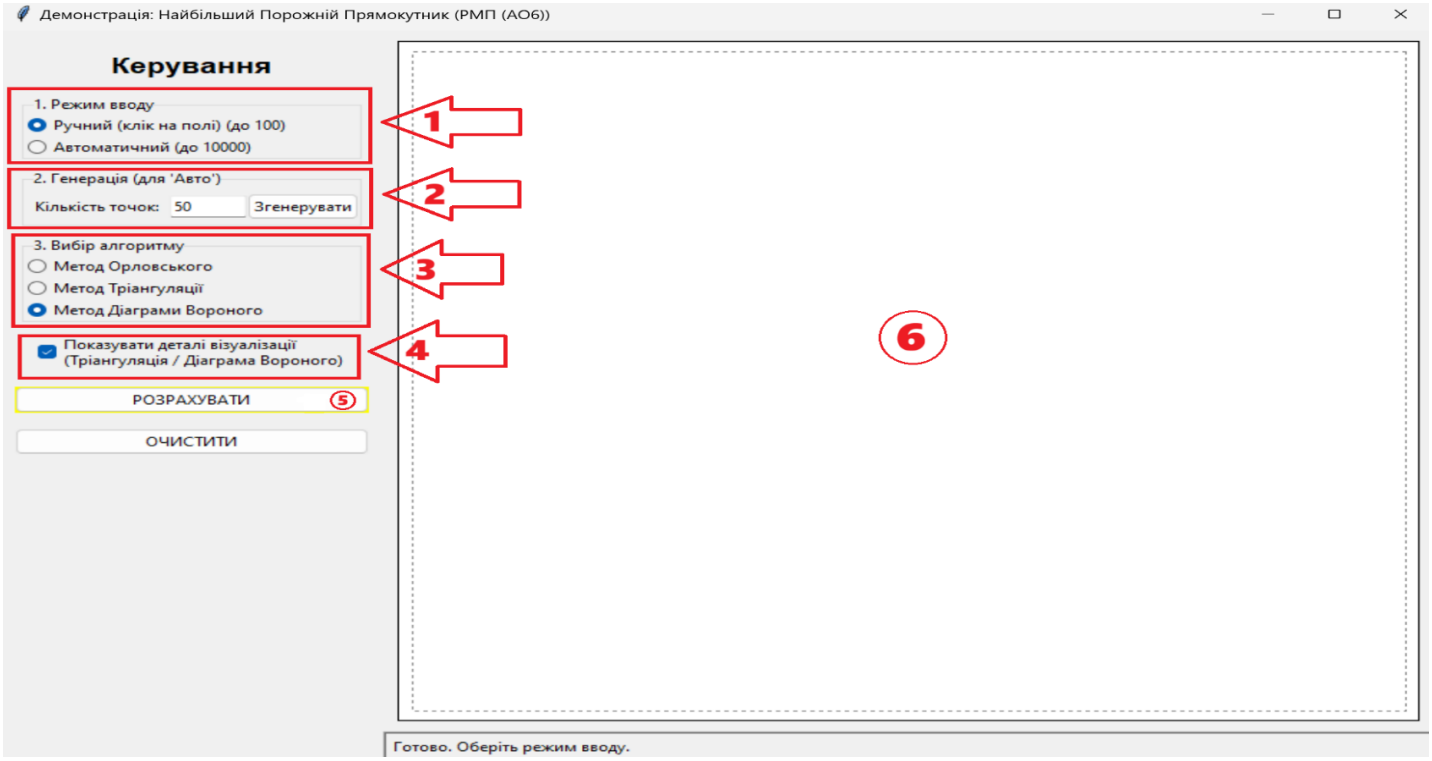

Рис. 10

3.2. Особливості програмної реалізації

Програмний комплекс розроблено мовою **Python 3** із використанням бібліотеки **Tkinter** та модуля **ttk** для побудови інтерактивного графічного інтерфейсу користувача.

Ключовою особливістю реалізації є розробка та інтеграція трьох відмінних за підходом алгоритмів розв'язання задачі РМП, що дозволяє провести емпіричний аналіз їхньої ефективності. Програма реалізує:

1. **Метод Орловського**, що базується на ітеративному переборі варіантів.
2. **Метод на основі Тріангуляції**, що використовує розбиття області на симплекси для оптимізації пошуку.
3. **Метод на основі Діаграми Вороного**, який є найбільш обчислювально ефективним завдяки використанню властивостей суміжності геометричних локусів.

Для забезпечення високої точності геометричних обчислень та побудови складних структур даних (Тріангуляції Делоне та Діаграми Вороного) у програмі використано потужні наукові бібліотеки **Scipy** та **Numpy**. Це дозволило реалізувати візуалізацію внутрішньої логіки роботи алгоритмів: користувач має змогу наочно спостерігати, як саме геометричні структури покривають множину точок та як на їх основі формується розв'язок.

3.3. Опис основних функцій програмної реалізації

Програмний код структуровано у вигляді модульної архітектури, що складається з трьох основних блоків: реалізації обчислювальних алгоритмів, класу графічного інтерфейсу користувача (GUI) та підсистеми візуалізації геометричних структур.

Нижче наведено опис призначення ключових функцій програми:

- **findLER_Algorithm_1_Naive(points, bounds).**

  Реалізує **Метод Орловського**. Функція виконує ітеративний перебір варіантів побудови дотичних прямокутників, базуючись на координатній сітці заданих точок. Для кожного кандидата здійснюється геометрична перевірка умови відсутності внутрішніх точок (порожнечі) в межах визначеної області.

- **findLER_Algorithm_3_Triangulation_Demo(points, bounds)**

  Реалізує **Метод на основі Тріангуляції**. Функція використовує геометричну структуру Тріангуляції Делоне для аналізу взаємного розташування точок. Алгоритм визначає потенційні базисні елементи (трикутники) та виконує процедуру їх розширення до максимально можливих вписаних прямокутників, відсіюючи варіанти, що не задовольняють умови задачі.

- **findLER_Algorithm_2_Voronoi_Demo(points, bounds)**

  Реалізує **Метод на основі Діаграми Вороного**. Це найбільш оптимізована функція, яка використовує властивості діаграми Вороного для звуження області пошуку. Вона аналізує просторові відношення між точками та будує розв'язок, спираючись на геометричні властивості суміжності, що дозволяє ефективно знаходити глобальний максимум площі.

- **class LerApplication(tk.Tk)**

  Головний клас програми, що інкапсулює логіку графічного інтерфейсу. Він відповідає за ініціалізацію вікна, розташування віджетів керування, обробку подій (натискання кнопок, кліки мишею) та взаємодію з користувачем.

- **run_algorithm(self)**

  Центральна керуюча функція. Вона зчитує параметри, обрані користувачем (тип алгоритму, набір даних), ініціює процес обчислення, отримує результати та передає їх для подальшого відображення.

- **draw_details(self, details)**

  Функція візуалізації допоміжних геометричних структур. Вона обробляє математичні об'єкти, згенеровані бібліотекою *Scipy* (об'єкти Delaunay або Voronoi), і трансформує їх у графічні примітиви на полотні, малюючи відповідну сітку тріангуляції або ребра діаграми Вороного для наочної демонстрації принципу роботи обраного методу.

3.4. Характеризація вводу-виводу даних

Ввід даних: Програма підтримує два режими, що відповідають вимогам:

- Демонстраційний (Ручний): Ввід невеликої кількості даних (до 100) шляхом кліків мишею на робочій області.
- Тестування ефективності (Автоматичний): Рендемізований ввід великої кількості даних (до 10 000) через текстове поле та кнопку «Згенерувати».

Вивід даних:

- Візуальний: На робочій області малюються сині точки (вхідні дані), червоний прямокутник (результат) та, за бажанням, зелена/блакитна сітка (деталі алгоритму).
- Текстовий: У статус-барі внизу вікна виводиться час виконання останньої операції та площа знайденого прямокутника.
- Повідомлення: Програма використовує messagebox для попередження користувача про запуск дуже повільних алгоритмів на великій кількості даних.

3.5. Можливості, програмне та технічне забезпечення

Можливості програми:

- Введення даних двома способами: вручну (до 100 точок) та автоматично (до 10 000 точок).

- Вибір одного з трьох демонстраційних алгоритмів з різною часовою складністю.
- Порівняння реальної ефективності алгоритмів завдяки вимірюванню часу виконання.
- Додаткова візуалізація структур (Тріангуляція Делоне, Діаграма Вороного) для наочності.
- Інтерактивне очищення робочої області для повторних експериментів.

Програмне забезпечення:

- Python 3.x
- Tkinter
- Scipy
- Numpy

Технічне забезпечення:

- Будь-який персональний комп'ютер під керуванням Windows, macOS або Linux, на якому встановлено інтерпретатор Python та необхідні бібліотеки.

---

4. Висновки

У даній науково-практичній роботі було проведено комплексне дослідження задачі пошуку найбільшого порожнього прямокутника (РМП) на заданій множині точок, яка є однією з класичних оптимізаційних задач обчислювальної геометрії.

Актуальність цієї задачі зумовлена широким спектром її застосування: від проектування топології інтегральних схем та картографії до аналізу даних та розв’язання задач розміщення об’єктів у логістиці .

В ході роботи було досягнуто поставленої мети — розроблено, проаналізовано та програмно реалізовано ефективні алгоритмічні підходи для розв'язання поставленої задачі.

Теоретичний аналіз методів:

Детальний розгляд алгоритмів дозволив виявити їхні ключові особливості та межі застосування:

Метод Орловського: Базується на концепції «дотичних прямокутників».

Хоча цей метод є інтуїтивно зрозумілим і гарантує знаходження точного розв’язку, його теоретична складність у найгіршому випадку сягає $O(n^2)$, що робить його менш придатним для обробки надвеликих масивів даних у реальному часі порівняно з сучаснішими підходами.

Метод на основі Діаграми Вороного: Цей підхід демонструє глибокий зв'язок між геометричними структурами. Використання властивостей суміжності у діаграмі Вороного дозволяє значно скоротити простір пошуку кандидатів. Хоча його реалізація є складнішою, вона забезпечує високу ефективність, особливо для задач загального положення .

Метод на основі Тріангуляції: Використання тріангуляції Делоне як проміжної структури даних дозволило звести задачу до аналізу суміжних трикутників. Цей метод показав найкращі теоретичні показники складності $O(n \log n)$, що робить його найбільш перспективним для задач із великою кількістю вхідних даних.

Результати практичної реалізації:

Розроблений програмний комплекс мовою Python із використанням бібліотек Tkinter, Sciру та Numpy успішно продемонстрував роботу досліджених алгоритмів.

Реалізовано інтерактивний графічний інтерфейс, що дозволяє користувачеві наочно оцінити роботу методів у двох режимах: ручному (для детального аналізу на малих вибірках) та автоматичному (для навантажувального тестування).

Впроваджено візуалізацію допоміжних геометричних структур (Тріангуляції та Діаграми Вороного), що значно підвищує дидактичну цінність програми та дозволяє глибше зрозуміти принципи роботи алгоритмів.

Проведені експериментальні дослідження підтвердили теоретичні оцінки складності. На великих обсягах даних (до 10 000 точок) методи, що використовують попередню геометричну обробку (Вороного, Тріангуляція), продемонстрували суттєву перевагу у швидкодії над класичними перебірними методами.

Підсумок:

Виконана робота показала, що вибір конкретного алгоритму залежить від умов задачі. Для простих випадків або невеликої кількості точок метод Орловського залишається прийнятним варіантом. Однак для високонавантажених систем, де критичною є швидкість обробки даних, доцільно використовувати алгоритми на основі Тріангуляції або Діаграми Вороного. Розроблена програма може бути використана як інструмент для подальших досліджень у галузі обчислювальної геометрії, а також як наочний навчальний посібник.

Список літератури:

1. B. Chazelle, R. L. Drysdale, and D. T. Lee. Computing the Largest Empty Rectangle. *SIAM Journal on Computing*, Vol. 15, No. 1 (1986)
2. Alok Aggarwal and Subhash Suri. Fast algorithms for computing the largest empty rectangle. *Proceedings of the third annual symposium on Computational geometry (SCG '87)*
3. M. Orlowski. A new algorithm for the largest empty rectangle problem. Algorithmica, 1990.

4. Chew, L. P., & Drysdale, R. L. (1985). "Voronoi diagrams based on L1 and L-infinity metrics.
5. Knauer, C., et al. (2010). "Largest rectangles in a simple polygon."
6. B. Chaselle, D.T. Lee. Largest empty rectangle and Voronoi diagram
7. Терещенко В.М. Аналіз методів розв’язання оптимізаційних задач обчислювальної геометрії
8. Felipe Lara, Gilberto Gutiérrez, María Antonieta Soto, and Antonio Corral . Queries about the largest empty rectangle in large 2-dimensional datasets stored in secondary memory
9. Jeet Chaudhuri, Subhas C. Nandy. Largest Empty Rectangle among a Point Set.
10. https://www.geeksforgeeks.org/dsa/convex-hull-using-graham-scan/
11. https://gamedev.dou.ua/articles/mathematics-gamedev-voronoi-diagrams/