Науково-практичний звіт на тему

ПОБУДОВА ОХОПЛЮЮЧОГО ПРОСТОГО МНОГОКУТНИКА НАЙМЕНШОЇ ПЛОЩІ

М. М. Осіпьонок, студент 4 курсу, групи ТК

Анотація. У роботі запропоновано евристичний алгоритм побудови простого многокутника найменшої площі, який би охоплював задану множину точок, вершинами якого є усі точки цієї множини. Алгоритм базується на рекурсивній побудові многокутників найменшої площі для підмножин точок та об'єднанні отриманих многокутників мінімізуючи загальну площу . Доведено часову складність алгоритму при $O(n^2)$ затратах пам'яті $O(n)$.

Abstract. In the paper we propose a heuristic algorithm for solving minimum area polygonization problem (MAP). The algorithm is based on recursive division of the set of points into two subsets, constructing an approximated solutions for the subsets and merging them with respect to minimizing the total area. Proved time complexity of an algorithm in $O(n^2)$ using $O(n)$ memory.

1 Вступ

Постановка проблеми. Задача апроксимації форми об'єкта за допомогою простіших об'єктів є важливою прикладною задачею обчислювальної геометрії. При цьому шукається апроксимація форми об'єкта, що є оптимальною в деякому розумінні, наприклад, має найменшу площу, найменший периметр і т.д. В роботі розглядається задача знаходження охоплюючого простого многокутника найменшої площі, для заданої множини точок, вершинами якого є всі точки множини. На практиці ця задача виникає в таких сферах як геоінформаційні системи [11], та для роботи з геосенсорною мережею [12]. В роботі [1] доведено NP-повноту поставленої задачі, тобто для знаходження точного розв'язку необхідно перебрати всі можливі прості многокутники, які можуть бути побудовані на заданій множині точок, яких може існувати експоненційно багато [2]. Тому в роботі запропоновано евристичний алгоритм розв'язання поставленої задачі.

Аналіз останніх досліджень. На сьогоднішній день для розв'язання задачі використовуються алгоритми апроксимації шуканого многокутника, побудовані на основі відомих алгоритмів генерації випадкового простого многокутника на заданій множині точок [3]. Для алгоритму *SteadyGrowth,* детальний опис якого можна знайти в роботі [3], використовують наступні критерії для відшукання локально оптимальних рішень, сподіваючись таким чином отримати розв'язок, близький до глобального оптимуму:

- Вибір початкового трикутника:
  1) Вибір випадкового трикутника
  2) Жадібний вибір трикутника
- Вибір допустимої точки
  1) Вибір випадкової точки з множини допустимих точок
  2) Жадібний вибір допустимої точки
- Вибір ребра
  1) Жадібний вибір ребра

В роботах [4, 5] детально описаний цей підхід до побудови апроксимованого розв'язку.

Також в роботі [5] доведено, що складність цього алгоритму є $O(n^3)$ при затратах пам'яті $O(n^2)$.

Новизна та ідея. В розглядуваній роботі запропоновано підхід, який дозволяє знайти наближений розв'язок задачі MAP з меншими затратами часу та пам'яті.

Мета статті. Розробити жадібний евристичний алгоритм розв'язання задачі MAP використовуючи підхід "Розділяй та перемагай".

2 Основна частина.

Сформулюємо геометричну постановку задачі MAP.

Постановка задачі MAP. Нехай задана множина *S* із *N* точок на площині. Для цієї множини визначити простий многокутник найменшої площі, який би охоплював задану множину точок, вершинами якого є усі точки множини *S*.

2.1. Побудова апроксимації розв'язку задачі MAP.

Якщо потужність множини |S| <= 5 , то знайдемо точний розв'язок простим перебором, інакше розділимо множину S на дві підмножини S_1 та S_2, потужність яких відрізняється не більше ніж на одиницю. Рекурсивно знайдемо розв'язок задачі для цих двох підмножин. Тепер, маючи апроксимації для множин S_1 та S_2 потрібно побудувати апроксимацію розв'язку для всієї множини S. Маємо 2 простих многокутники, що не перетинаються. Для об'єднання їх в один многокутник виберемо два ребра e_1 з S_1 та e_2 з S_2 . Позначимо AB=e_1 та CD=e_2 тоді для того щоб в результаті об'єднання двох простих многокутників отримати простий многокутник, необхідно щоб відрізки BC та AD перетинали многокутники лише в вершинах: A,B,C,D. Тоді вилучивши ребра e_1, e_2 та додавши нові ребра BC та AD отримуємо простий многокутник. Для мінімізації загальної площі доцільно вибрати ребра e_1, e_2 таким чином, щоб площа чотирикутника ABCD була б найменшою. Розглянемо тепер питання "яким чином вибирати ребра e_1 та e_2?" Позначимо CH(S_1) та CH(S_2) – опуклі оболонки для відповідних підмножин. Для опуклих оболонок знайдемо верхню та нижню дотичні (опорні) прямі (наприклад, алгоритмом, описаним в [6]). Якщо точки многокутника задані в порядку обходу проти годинникової стрілки, то для многокутника, що знаходиться лівіше, підланцюг від точки перетину нижньої дотичної до верхньої позначимо l_1. Для іншого многокутника, підланцюг від точки перетину верхньої дотичної прямої до нижньої позначимо l_2. Тоді ребро e_1 належить l_1, а ребро e_2 належить l_2, оскільки відрізок проведений з будь-якої точки S_1, що не належить l_1 обов'язково перетне хоча б один відрізок з l_1, аналогічно і для S_2.

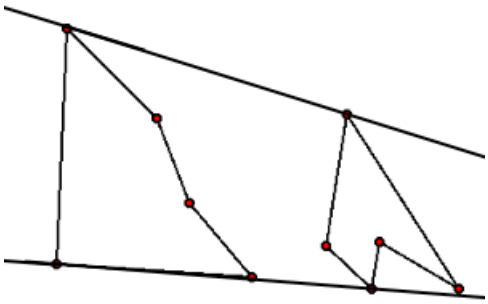


Рисунок 1. Верхня та нижня дотичні прямі

Таким чином, залишилось перебрати пари ребер з l_1 та l_2, та серед тих пар, з'єднавши які отримуємо коректний чотирикутник, вибрати ту, яка задає чотирикутник найменшої площі. Це можна реалізувати таким чином: Позначимо L – множина ребер, які входять до підланцюгів l_1 та l_2. Тоді виберемо один з ланцюгів, без обмежень загальності вважаємо, що вибрали l_1. Для кожного відрізка з l_1 (яких може бути $O(n)$) знайдемо множину видимих точок (вершин) з множини L. Для одного відрізка це можна зробити за час, лінійний по кількості елементів в L, тобто $O(n)$ [7]. Тоді, якщо вершини відрізку, що належить l_2 є видимими з вершин певного відрізку з l_1 то ця пара відрізків задає коректний чотирикутник. Для всіх відрізків з l_1 можна знайти множину видимих відрізків з l_2 за час $O(n^2)$.

Алгоритм.

Вхід: Множина S з n точок на площині.
Вихід: Многокутник, що апроксимує розв'язок задачі MAP.

Попередня обробка: відсортувати точки множини по x – координаті.
1. Якщо |S| <= 5, то повернути точний розв'язок, знайдений перебором.
Інакше розділити множину S на дві частини S_1 та S_2. Знайти розв'язок для цих підмножин.
2. Об'єднати розв'язки для підмножин S_1 та S_2 алгоритмом MergeSimplePolygons і отримати, таким чином, розв'язок для всієї множини S.

Алгоритм MergeSimplePolygons:

Вхід: Два простих многокутника P_1 та P_2 .
Вихід: Простий многокутник, отриманий об'єднанням P_1 та P_2 .

1. Знайти опуклі оболонки для многокутників P_1 та P_2 : $CH(P_1)$ та $CH(P_2)$.
2. Знайти верхню та нижню дотичні прямі для опуклих многокутників $CH(P_1)$ та $CH(P_2)$.
3. Будуємо підланцюги l_1 та l_2 .
4.Для кожного ребра з l_1 :
Знаходимо множину видимих вершин з множини L
Для кожного видимого ребра:
Знаходимо площу чотирикутника, який задає відповідна пара ребер, і якщо черговий чотирикутник має мінімальну площу, то запам'ятовуємо його.
5. З'єднуємо многокутники, вилучаючи два зайвих ребра.

3 Обґрунтування складності

Теорема 1. Складність наближеного розв'язання задачі MAP становить $O(n^2)$ операцій за умови, що $O(nlogn)$ операцій піде на попередню обробку.

Доведення. Попередня обробка точок множини S – сортування точок цієї множини, яке можна виконати будь-яким оптимальним алгоритмом сортування за $O(nlogn)$ [8].
Оскільки алгоритм використовує стратегію "розділяй та володарюй", то якщо складність операції злиття є $O(f(n))$, загальна складність визначається розв'язком співвідношення $T(n)=2T(\frac{n}{2})+f(n)$. Залишилось оцінити функцію $f(n)$. Знаходити опуклу оболонку доцільно використовуючи стратегію "розділяй та володарюй", підтримуючи опуклі оболонки для двох многокутників, злиття яких виконується. Об'єднати дві опуклі оболонки можна за $O(n)$ [9]. Побудова дотичних прямих відбувається під час злиття опуклих оболонок. Побудова підланцюгів l_1 та l_2 – прохід по контуру відповідних многокутників, тому, вимагає $O(n)$ операцій. В пункті 2.1 описано алгоритм знаходження чотирикутника мінімальної площі, яким можна об'єднати два простих многокутника. Знайти видимі вершини (з відрізка) можна за час $O(n)$ [7], цю операцію необхідно виконати для всіх відрізків одного з підланцюгів, тому загальна складність відшукання чотирикутника найменшої площі складає $O(n^2)$. Таким чином, складність операції merge складає $O(n^2+n)=O(n^2)$ операцій. Тобто, маємо співвідношення $T(n)=2T(\frac{n}{2})+n^2$. Розв'язком співвідношення, згідно з теоремою [10] є $O(n^2)$.

Теорема 2. Для роботи описаного вище алгоритму з складністю, описаною в умові теореми 1, необхідно $O(n)$ пам'яті.

Доведення. Для сортування необхідно не більше $O(n)$ пам'яті (залежно від обраного алгоритму). Для підтримки опуклої оболонки протягом роботи алгоритму необхідно, також, $O(n)$ пам'яті [9]. Залишається показати, що для відшукання чотирикутника мінімальної площі необхідно $O(n)$ пам'яті. Справді, для реалізації наведеного алгоритму достатньо сформувати два списки l_1 та l_2, довжина яких є $O(n)$ та два вказівники на елементи з першого та другого списку. Для пошуку видимих відрізків необхідно $O(n)$ пам'яті. Оскільки для кожного відрізка з l_1 не потрібно підтримувати множину видимих відрізків для решти відрізків з l_1 то і загальні затрати пам'яті на виконання алгоритму складають $O(n)$.

4 Практична частина

Розроблено реалізацію описаного алгоритму на мові C++ з використанням фреймворку Qt 5.10. Програма підтримує два способу задання точок — ручний ввід, та автоматична генерація заданої кількості випадкових точок.

Основні функції програмної реалізації

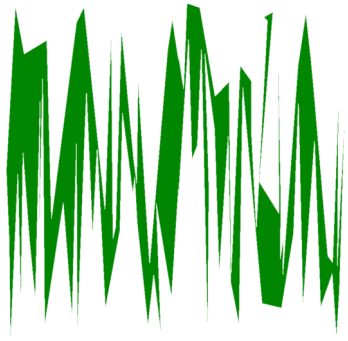
`void` **`brute_optimal_polygon`**`(Polygon & v, int l, int r)` – перебором визначає многокутник найменшої площі на множині вершин [l,r] .

`void` **`minimum_quadrilateral_visibility`**`(vector<pair<Point, Point>> & seg_l, vector<pair<Point, Point>> & seg_r, pair<Point, Point> & upperTangent, pair<Point, Point> & lowerTangent, pair<Point, Point> & u1, pair<Point, Point> & u2)` - знаходить чотирикутник найменшої площі між двома ланцюгами використовуючи алгоритм видимості [7], результат знаходиться в u1 та u2.

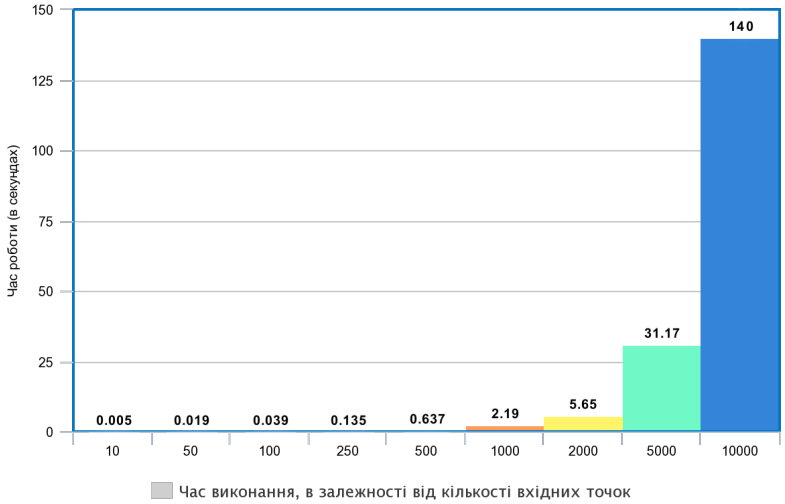
`Polygon` **`merge_polygons`**`(Polygon & v, int l, int r, int mid, Polygon & lhull, Polygon & rhull)` – зливає два простих многокутника, та повертає опуклу оболонку отриманого многокутника.'

Приклади згенерованих програмою многокутників наведені на рисунках нижче.

30 точок



100 точок



5 Висновки

У роботі запропоновано евристичний алгоритм розв'язання задачі "мінімальний охоплюючий простий многокутник", що використовує стратегію "розділяй та володарюй". Наведений алгоритм можна використовувати в різноманітних прикладних областях, де виникає задача MAP, зокрема в розпізнаванні образів.

Доведено обчислювальну складність $O(n^2)$ алгоритму, при витратах $O(nlogn)$ на попередню обробку та затратах пам'яті $O(n)$, з цього слідує, що наведений алгоритм є одним із найоптимальніших алгоритмів для знаходження апроксимації розв'язку вказаної вище задачі. Покращення точності роботи алгоритму можливе за допомогою декількох запусків алгоритму на видозміненій множині точок, наприклад, повернути точок на певний кут, що еквівалентно розділенню множини точок прямою, що не паралельна осі ординат. Визначення оптимальної кількості ітерацій потребує додаткового дослідження.

Список літератури

1. *S. P. Fekete.* On Simple Polygonalizations with Optimal Area // Department of Mathematics, TU Berlin, Str. des 17. Juni 136, D-10623 Berlin, Germany
2. Jeff Erickson Generating random simple polygons // http://jeffe.cs.illinois.edu/open/randompoly.html
3. Thomas Auer, Martin Held Heuristics for the Generation of Random Polygons // Extended abstract in Proc. 8th Canad. Conf. Comput. Geometry, pp. 38-44, 1996.
4. Maria Teresa Taranilla, Edilma Olinda Gagliardi, Gregorio Hernandez Penalver Approaching Minimum Area Polygonization // Facultad de Ciencias Fısico, Matematicas y Naturales Universidad Nacional de San Luis, Argentina
5. V. Muravitskiy, V. Tereshchenko Generating a simple polygonalizations // 2011 15th International Conference on Information Visualisation
6. Tangents to & between 2D Polygons // http://geomalgorithms.com/a15-_tangents.html
7. B. Joe and R.B. Simpson. Corrections to lee's visibility polygon algorithm
8. Donald E. Knuth Sorting and Searching, The Art of Computer Programming, 3 (2nd ed.), Boston: Addison-Wesley, ISBN 0201896850
9. Preparata, F.P. and Hong, S.J. Convex Hulls of Finite Sets of Points in Two and Three Dimensions. // Communications of the ACM, 20, 87-93. // 1977
10. The Master Theorem // https://math.dartmouth.edu/archive/m19w03/public_html/Section5-2.pdf
11. H. J. Miller and J. Han, Eds., Geographic Data Mining and Knowledge Discovery. // CRC Press, 2001
12. M. F. Worboys and M. Duckham, “Monitoring qualitative spatiotemporal change for geosensor networks,” // International Journal of Geographic Information Science, vol. 20, no. 10, pp. 1087–1108, 2006.

Додатки

1. https://github.com/osipyonok/OGKG_Lab/tree/master/OGKG_Lab_Gui – Код розробленої програми