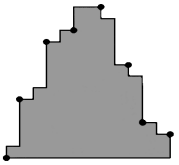
Науково-практичний звіт на тему
ЗАДАЧА ОРТОГОНАЛЬНОГО ТЮРЕМНОГО ДВОРУ

В. Р. Тунік, студент 3 курсу, групи комп'ютерна математика 1

Анотація. У роботі запропоновано оригінальний алгоритм розміщення відеокамер за допомогою тріангуляції Делоне та жадібного алгоритму в задачі ортогонального тюремного двору. Застосування цього алгоритму до відповідної задачі.



Abstract. In this paper, we propose an original algorithm for video camera placement using Delaunay triangulation and greedy algorithm in the orthogonal prison yard problem and apply this algorithm to the corresponding problem.

1 Вступ

Постановка проблеми. Задача про ортогональне тюремне подвір'я є різновидом задачі про вартових в обчислювальній геометрії. Задача полягає в тому, щоб розмістити охоронців (відеокамери) у різних точках простого полігона (що представляє стіни в'язниці) таким чином, щоб кожна точка внутрішньої і зовнішньої області була охоплена хоча б одним охоронцем (відеокамерою).

Задача має практичне застосування у сфері безпеки, де вона може бути використана для визначення мінімальної кількості відеокамер, необхідної для захисту заданої території. Задача також вивчалася в контексті художніх галерей [1], де вона використовується для визначення мінімальної кількості відеокамер, необхідних для охорони музею або художньої галереї.

Задача має теоретичне значення в обчислювальній геометрії, де вона використовується для вивчення складності алгоритмів для задач видимості. Розв'язок задачі може бути корисним при розробці ефективних алгоритмів для інших споріднених задач обчислювальної геометрії.

Аналіз останніх досліджень. Проблема ортогонального тюремного двору залишається важливою проблемою в обчислювальній геометрії. Оригінальна проблема художньої галереї, яку підняв V. Klee запитує, скільки охоронців достатньо для того, щоб наглядати за внутрішньою частиною n-стороннього простого многокутника. 1975 V. Chvatal надав відповідь, яка доводить, що $\lfloor n/3 \rfloor$ охоронців(відеокамер) завжди достатньо, а іноді і необхідно [2].

Задача про тюремний двір для загальних простих многокутників була повністю розв'язана Z Furedi та D Kleitma, які довели, що $\lceil n/2 \rceil$ вершин з відеокамерами для опуклих $\lfloor n/2 \rfloor$ вершин з відеокамерами для довільного не опуклого простого многокутника є достатніми. [3]. В той же час Frank Homann та Klaus Kriegel опублікували нову границю: $\lfloor 5n/16 \rfloor+2$ для ортогонального опуклого тюремного двору. Згодом ця межа була покращена до $\lfloor 5n/12 \rfloor + 1$ в роботі T.S. Michael та V. Pinciu. [4]

Одним з алгоритмів розв'язання задачі розміщення відеокамер у задачі ортогонального тюремного двору є жадібний алгоритм. [5] Цей алгоритм базується на спостереженні, що охоронці повинні бути розміщені у вершинах многокутника. Алгоритм починає з вибору довільної вершини багатокутника і розміщення на ній охоронця. Потім він видаляє всі вершини, які є видимими для цього охоронця. Алгоритм повторює цей процес до тих пір, поки всі вершини не будуть видимі охоронцями. Алгоритм має часову складність O(n^2), де n - кількість вершин у многокутнику.

Новизна та ідея. В цій роботі запропоновано покращений загальний алгоритм для розв'язання задачі ортогонального тюремного двору, використовуючи тріангуляцію та жадібний алгоритм.

Мета статті. Розробити узагальнений алгоритм розміщення камер в задачі ортогонального тюремного двору.

2 Основна частина

Задача про тюремне подвір'я є однією з сімейства задач про охорону, де потрібно розмістити охоронців (відеокамери) у різних точках або на простому багатокутнику (що представляє стіни в'язниці) з метою охопити (бачити) кожну точку внутрішнього та зовнішнього подвір'я принаймні одним охоронцем (відеокамерою).

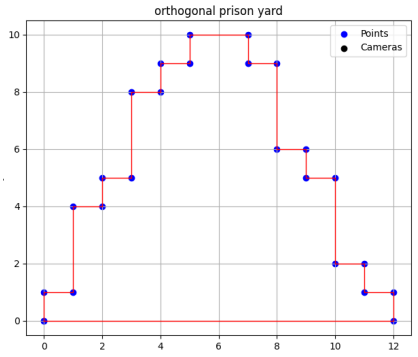
Означення 1. *Охоплення (або бачення) точки означає наявність безперешкодного прямого огляду від якогось охоронця (вершини з відеокамерою) до іншої відеокамери. Дві точки називаються видимі одна з одної, якщо відрізок, що їх з'єднує, не перетинається з жодною перешкодою.* [6]

Теорема 1. Якщо обраний набір вершин охоплює всі вершини багатокутника, то такий набір охоплює і всі його сторони.

Доведення. Підемо від супротивного. Припустимо, що існує набір вершин, котрий охоплює всі сторони багатокутника, але не охоплює всіх його вершин. Тобто існує принаймні одна вершина, і як наслідок одна сторона, до якої належить ця вершина, яку не охоплює наш набір вершин. Суперечність. Доведено.

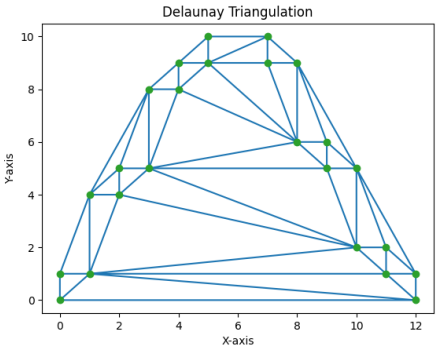
Постановка задачі: Розставити мінімальну кількість відеокамер у вершинах так, щоб уся територія двору (усі вершини багатокутника, що утворює двір) була видима.

2.1 Побудова розв'язку задачі.



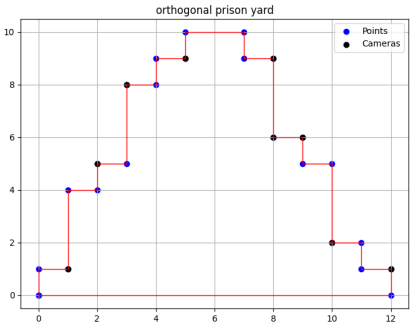
Нехай маємо, деякий набір точок, що утворює ортогональний політоп (Наприклад: {(0, 0), (0, 1), (1, 1), (1, 4), (2, 4), (2, 5), (3, 5), (3, 8), (4, 8), (4, 9), (5, 9), (5, 10), (7, 10), (7, 9), (8, 9), (8, 6), (9, 6), (9, 5), (10, 5), (10, 2), (11, 2), (11, 1), (12, 1), (12, 0)})

Виконаємо тріангуляцію Делоне [7] полігону з прикладу. Очевидно, що сусідні точки в тріангуляції - видимі. Отже, вершині, що відповідає найбільша кількість трикутників - відповідає найбільша кількість видимих вершин.



Тепер обираємо вершини куди розмістити відеокамеру за критерієм того, що вона охоплює найбільшу кількість нових (раніше не охоплених) вершин, тобто належить найбільшій кількості трикутників. Запам'ятовуємо цю вершину, а трикутники, в яких вона лежить, вилучаємо з нашого списку трикутників тріангуляції. Повторюємо ці дії допоки усі вершини стануть охопленими.

2.2 Алгоритм



На вхід подається набір точок у вигляді “x, y”, що утворюють ортогональний політоп.

1. Виконуємо тріангуляцію Делоне полігону за допомогою алгоритму Quickhull [8] для заданих точок. Записуємо утворені трикутники в масив з наборами з 3 точок відповідних трикутників: [[(x11,y11),(x12,y12),(x13,y13)],[(x21,y21),(x22,y22),(x23,y23)], …].

Далі використаємо отриманий масив трикутників для знаходження вершин, куди розмістимо камеру за принципом жадібного алгоритму.

2. Зведемо масив трикутників в єдиний масив відповідних точок (пройдемося по кожному трикутнику і запишемо його точки в масив) і знайдемо найчастішу вершину. Таким чином ми знайшли вершину, що зустрічається найчастіше серед усіх трикутників.
3. Видаляємо всі трикутники, які містять знайдену вершину (проходимся по всім трикутникам і додаємо трикутник в новий масив трикутників, якщо даний не містить точки, що ми знайшли на другому кроці).
4. Тепер масивом трикутників для кроків 2-3 є новий масив, утворений в результаті 3-го кроку. Повторюємо кроки 2-3 допоки новий масив трикутників не буде порожнім.

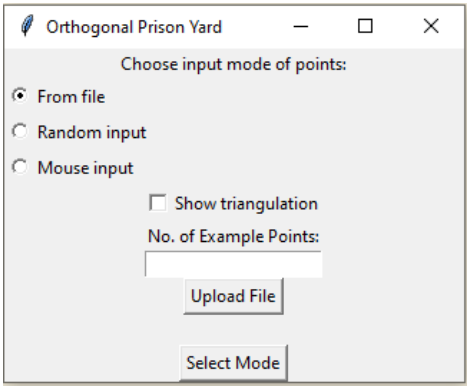
3 Обґрунтування складності

Теорема 2. Часова складність розв’язання задачі розміщення мінімальної кількості відеокамер у вершинах так, щоб уся територія двору була видима становить *O(N logN)*.

Доведення. 1 крок: тріангуляції Делоне за допомогою ефективного алгоритму Quickhull, має часову складність O(n log n) [8]. 2 крок: на цьому кроці список трикутників перетворюється на єдиний список точок. Часова складність зведення списку трикутників становить O(n). Знаходження точки, що найчастіше зустрічається має часову складність O(1). Звідси 2 крок має складність O(n). 3 крок: видалення трикутників, що містять точку, яка найчастіше зустрічається передбачає ітерацію по кожному трикутнику, тобто O(n) та перевірку на наявність точки, що найчастіше зустрічається у кожному трикутнику O(1). Тому цей крок має складність O(n). 4 крок: так як на кожній ітерації ми обираємо найкращий варіант, а складність кроків 2-3 дорівнює O(n), то часова складність рівна O(n log n).

Отже, робимо висновок, що алгоритм має часову складність O(n log n).

4 Практична частина



Особливістю даної реалізації є можливість побачити проміжний етап тріангуляції та кінцеве розміщення відеокамер на ортогональному тюремному подвір’ї.

На малюнку проілюстрований програмний інтерфейс.

Для вводу даних доступні 3 режими: з файлу, випадкова генерація, за допомогою мишки. Реалізація алгоритму виконана за допомогою мови програмування Python та її бібліотек: matplotlib - для виводу графічного результату в функціях create_polygon_figure та delaunay_triangulation, tkinter - для інтерфейсу, scipy - для виконання тріангуляції Делоне [9] та collections в проміжних етапах алгоритму. Приклад виводу графічного розв'язку зображено в основній частині статті. Реалізація за допомогою мишки базується на реалізації Pedro Correia (також відомий як armatita) [10]

Лістинг функції коду, що реалізує основний алгоритм, за допомогою допоміжних функцій, лістинг яких не ілюструємо:

```
def main():
    mode_choice = mode_var.get()

    if mode_choice == '1':
        filename = "points.txt"
    elif mode_choice == '2':
        n = int(entry_points.get())
        create_random_points(n)
        filename = "random_points.txt"
    elif mode_choice == '3':
        mouse_input.create_points()
        filename = "mouse_points.txt"

    build_fig = 1 if build_var.get() == 1 else 0

    points = read_points_from_file(filename)
    triangulation_points = delaunay_triangulation(points, build_figure=build_fig)

    cameras = []
    triangles_without_most_common = triangulation_points[:]
    while triangles_without_most_common:
        all_points = [point for triangle in triangles_without_most_common for point in triangle]
        point_counter = Counter(all_points)
        most_common_point, occurrences = point_counter.most_common(1)[0]
        cameras.append(most_common_point)
        triangles_without_most_common = [triangle for triangle in triangles_without_most_common
                                         if most_common_point not in triangle]

    create_polygon_figure(points, cameras)
```

5 Висновки

Розробка алгоритму розміщення відеокамер з використанням тріангуляції Делоне та жадібного підходу для ортогональної задачі тюремного двору є значним внеском в оптимізацію системи відеоспостереження.

Протягом роботи над програмною реалізацією було проведена велика кількість експериментів (див. Додатки) , які підтверджують ефективність алгоритму та відповідності верхній межі кількості відеокамер $\lfloor 5n/12 \rfloor + 1$ з роботи T.S. Michael та V. Pinciu.

Запропонований алгоритм має сильні сторони: використовує тріангуляцію Делоне - надійний метод для ефективної дискретизації простору; жадібний підхід - практичний для оптимізації розміщення камер на основі певних критеріїв; ефективно вирішує проблему ортогональних тюремних дворів. Проте присутні недоліки: значно більші двори можуть призвести до збільшення обчислювальної складності; обробка змін у будові двору потребує виконання алгоритму спочатку.

Перспективи та вдосконалення: 1. оптимізація - дослідити методи підвищення ефективності алгоритму, такі як розпаралелювання або евристики для зменшення

обчислювальних накладних витрат; 2. динамічна адаптація - підвищення адаптивності алгоритму до змін середовища або макетів шляхом впровадження динамічних стратегій перепозиціонування камери; 3. алгоритмічні вдосконалення - альтернативні методи тріангуляції або комбіновані алгоритми для кращої точності та покриття.

Подальший розвиток: 1. адаптація в реальному часі - дослідити підходи для адаптації в реальному часі до динамічних ситуацій в середовищах спостереження; 2. інтеграція машинного навчання - дослідити інтеграцію моделей машинного навчання для прогнозування потенційних сліпих зон або оптимізації розміщення камер на основі історичних даних.

Отже, алгоритм розміщення камер, заснований на тріангуляції Делоне та жадібних стратегіях, є міцною основою для ефективної оптимізації відеоспостереження. Покращення масштабованості, адаптивності до динамічних змін та оптимізація обчислювальної ефективності є ключовими напрямками для подальшого розвитку та удосконалення алгоритму та програмної реалізації. Проведення експериментів підтверджує ефективність алгоритму за різних умов.

Список літератури:

[1]https://en.wikipedia.org/wiki/Art_gallery_problem

[2]https://www.sciencedirect.com/science/article/pii/0095895675900611

[3] Zoltán Füredi, D. J. Kleitman, “The prison yard problem”, 1994

[4] T.S. Michael and V. Pinciu. Guarding orthogonal prison yards: An upper bound. Congr. Numerantium, 211:57–64, 2012.

[5] J. O’Rourke. “Computational Geometry in C”. Cambridge University Press, Second Edition, September 1998.

[6]https://en.wikipedia.org/wiki/Visibility_(geometry)

[7]https://www.cs.cmu.edu/~quake/triangle.defs.html

[8]https://dpd.cs.princeton.edu/Papers/BarberDobkinHuhdanpaa.pdf

[9]https://docs.scipy.org/doc/scipy/reference/generated/scipy.spatial.Delaunay.html

[10]https://stackoverflow.com/questions/37363755/python-mouse-click-coordinates-as-simply-as-possible

[11]https://medium.com/@srosamazaid/the-greedy-algorithm-pattern-an-in-depth-analysis-7bb28d5dbfa7

Додатки

github: https://github.com/duinage/prison_yard_problem

Деякі результати експериментів з розміщення камер для різних наборів точок:

```
all points amount, n =  24
cameras points amount =  9
upper bound for cameras ⌊5n/12⌋+1 =  11
all points amount, n =  72
cameras points amount =  25
upper bound for cameras ⌊5n/12⌋+1 =  31
all points amount, n =  76
cameras points amount =  30
upper bound for cameras ⌊5n/12⌋+1 =  32
all points amount, n =  46
cameras points amount =  17
upper bound for cameras ⌊5n/12⌋+1 =  20
all points amount, n =  18
cameras points amount =  6
upper bound for cameras ⌊5n/12⌋+1 =  8
all points amount, n =  28
cameras points amount =  11
upper bound for cameras ⌊5n/12⌋+1 =  12
all points amount, n =  100
cameras points amount =  40
upper bound for cameras ⌊5n/12⌋+1 =  42
```