## Науково-практичний звіт на тему

**ЗАДАЧА ОРТОГОНАЛЬНОГО ТЮРЕМНОГО ДВОРУ**

**Володимир Кушнір, студент 3 курсу, група ТК-31**

### Анотація

У роботі розглянуто задачу оптимального розміщення камер спостереження у внутрішньому дворі, що моделюється ортогональним (прямокутним) багатокутником. Запропоновано програмну реалізацію трьох підходів до покриття багатокутника камерами — жадібного, геометричного евристичного та на основі тріангуляції Делоне. Здійснено візуалізацію результатів та порівняння ефективності методів. Програма має графічний інтерфейс, який дозволяє завантажувати власні багатокутники або генерувати випадкові.

### Abstract

The paper addresses the problem of optimal placement of surveillance cameras inside a courtyard modeled as an orthogonal polygon. We propose a software implementation of three camera placement strategies — greedy, geometric heuristic, and Delaunay triangulation. The system visualizes and compares the effectiveness of each algorithm. A graphical user interface allows users to either upload polygon vertex data or generate random polygons.

## 1 Вступ

### *Постановка проблеми.* У роботі досліджується задача ефективного розміщення камер спостереження в середині внутрішнього двору, який моделюється ортогональним багатокутником. Завдання полягає в мінімальному покритті всієї області полігону шляхом розташування обмеженої кількості камер таким чином, щоб кожна точка периметра була видимою принаймні з однієї камери.

Ця задача має практичне застосування в охоронних системах, проєктуванні приміщень, а також у теоретичній інформатиці як одна з форм задачі охоплення множини з геометричними обмеженнями.

### *Актуальність.* Оскільки оптимальне розміщення камер у двовимірному просторі з перешкодами належить до класу NP-складних задач, особливу увагу приділено евристичним методам. У даній роботі реалізовано та проаналізовано три таких підходи. Враховуючи зростаючі вимоги до автоматизованого проектування систем відеоспостереження в публічних і приватних просторах, дослідження даної задачі є важливим як з практичної, так і з теоретичної точки зору.

### *Аналіз попередніх досліджень.* Задача охоплення полігону відома в обчислювальній геометрії як "Art Gallery Problem", і у випадку ортогонального полігону вона допускає ефективні апроксимації. Існують алгоритми, що гарантують розв'язок у межах ⌊n/4⌋ камер для простого ортогонального багатокутника з n вершинами. Проте, побудова таких алгоритмів не тривіальна, і часто вдаються до жадібних чи евристичних стратегій, які забезпечують прийнятну якість результату.

### *Мета роботи.* Метою роботи є розробка програмного забезпечення, яке реалізує і порівнює кілька алгоритмів розміщення камер у внутрішньому ортогональному полігоні, а також оцінка їх продуктивності та ефективності з точки зору кількості необхідних камер.

## 2. Основна частина

### Геометрична постановка задачі

#### Постановка задачі покриття полігона камерами. Нехай задано ортогональний багатокутник PPP, який моделює умовний "тюремний двір". Його вершини задані як множина точок , з'єднаних у замкнений полігон. Необхідно знайти найменшу множину точок C⊆S, у які треба розмістити камери спостереження так, щоб кожна точка S була видимою з хоча б однієї камери.

**Означення 1.** Кажуть, що точка q є видимою з точки p, якщо відрізок  ​ повністю лежить всередині або на межі полігона P і p та q лежать або на одній горизонталі, або на одній вертикалі (горизонтальна або вертикальна видимість без перетину зі стінками полігона).

### 2.1 Алгоритмічна побудова розв’язків

У рамках роботи реалізовано три методи вирішення задачі:

#### Метод 1: Жадібний алгоритм (Greedy)

Жадібний підхід обирає вершину, з якої видно найбільше непокритих точок. На кожному кроці алгоритм оновлює множину ще не покритих точок і повторює процес.

def place\_cameras\_greedy(points):

**Переваги:** швидкий, простий у реалізації, добре працює на практиці.  
**Недоліки:** не гарантує мінімальної кількості камер.

#### Метод 2: Тріангуляція Делоне

Алгоритм будує тріангуляцію Делоне для множини вершин, після чого вибирає точки, які покривають найбільше трикутників. Трикутники, що вже покриті, виключаються з подальшого розгляду.

def place\_cameras\_delaunay(points):

**Переваги:** базується на відомій геометричній структурі, що добре працює для довільних полігонів.  
**Недоліки:** іноді розміщує надлишкові камери через надмірне покриття.

#### Метод 3: Геометрична евристика

Кожній вершині полігона призначається рейтинг — кількість вершин, видимих з неї. Алгоритм обирає вершину з найвищим рейтингом, додає її до списку камер і виключає всі покриті точки. Повторюється доти, поки всі точки не покрито.

def place\_cameras\_geometry(points):

**Переваги:** ефективний у спеціалізованих (наприклад, ортогональних) полігонах.  
**Недоліки:** може потребувати додаткових перевірок для складних форм.

### 2.2 Формальна постановка задачі покриття

Нехай:

* V — множина всіх вершин полігона;
* Vis(pi​)⊆V — підмножина вершин, видимих з вершини pi​;
* U⊆V — непокриті вершини;
* C⊆V — розташування камер.

Задача полягає у побудові C, де ⋃c∈CVis(c)=V, і ∣C∣→min.

Це — узагальнена задача покриття множини, яка є NP-повною, тому застосовуються евристики (жадібний вибір, геометричний аналіз, тріангуляція).

### 2.3 Реалізація

У програмі реалізовано:

* графічний інтерфейс на **Tkinter**;
* побудову випадкових ортогональних полігонів;
* завантаження власних точок із .txt файлу;
* інтерактивне порівняння трьох методів на одному полігоні.

Головні етапи роботи програми:

1. Зчитування точок із файлу або генерація.
2. Побудова полігона.
3. Застосування трьох методів.
4. Візуалізація результатів — окремо для кожного методу.

Графік будується з допомогою **matplotlib**, для геометричних операцій використано **scipy.spatial.Delaunay**, для підрахунків — **collections.Counter**.

## 3. Аналіз складності алгоритмів

У даному розділі розглянемо часову складність трьох реалізованих алгоритмів.

### 3.1 Жадібний алгоритм (Greedy)

На кожній ітерації обирається вершина з найбільшою кількістю ще непокритих видимих точок.

* Побудова карти видимості: O(n^2)
* Кожна ітерація: O(n)
* Загальна складність: **O(n^2)** у найгіршому випадку.

**Переваги:** просто реалізується, швидко працює на малих полігонах.  
**Недоліки:** не завжди знаходить оптимальну кількість камер.

### 3.2 Тріангуляція Делоне

* Побудова тріангуляції: O(nlogn)
* Прохід по трикутниках для вибору точок: O(n)
* Загальна складність: **O(nlogn)**

**Переваги:** добре масштабується на більших вхідних даних.  
**Недоліки:** не враховує геометричну структуру полігона, покриття точок не завжди оптимальне.

### 3.3 Геометрична евристика

* Перевірка видимості кожної пари: O(n^2)
* Оцінка найкращої точки для розміщення: O(n)
* Загальна складність: **O(n^2)**

**Переваги:** враховує тільки дозволену видимість (по вертикалі та горизонталі), підходить до специфіки ортогональних полігонів.  
**Недоліки:** складніша в реалізації логіка порівняно з greedy.

## 4. Експериментальні результати

Для демонстрації роботи було проведено серію тестів на полігонах з 10, 20, 30, 40 і 50 вершинами. В результаті порівняння кількості камер в середньому:

|  |  |  |  |
| --- | --- | --- | --- |
| **Кількість вершин** | **Greedy** | **Delaunay** | **Geometry** |
| 10 | 3 | 4 | 2 |
| 20 | 5 | 6 | 4 |
| 30 | 7 | 8 | 6 |
| 40 | 9 | 10 | 7 |
| 50 | 11 | 12 | 8 |

**Висновок**: геометрична евристика демонструє найкращі результати щодо кількості необхідних камер, але потребує більше часу на розрахунок видимості. Жадібний підхід і тріангуляція працюють швидше, але менш точно.

## 5. Висновки

У роботі було реалізовано та порівняно три евристичних методи розв’язання задачі мінімального розміщення камер у замкнутому ортогональному багатокутнику:

* **Greedy**: швидкий, але менш точний;
* **Delaunay**: геометрично зручний, але дає зайві точки;
* **Geometry heuristic**: найточніший у специфічних випадках (прямолінійна видимість).

Програма дозволяє зчитувати вхідні дані з файлу, генерувати нові полігони та порівнювати методи на практиці. Отримані результати демонструють важливість геометричної евристики для точних і оптимізованих рішень задач візуального контролю територій.

Список літератури:

[1] Шевченківська весна – 2017, ст. 143.-146.

[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:ст. 57–64, 2012.

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

[6]https://www.cs.cmu.edu/~quake/triangle.defs.html [7]https://dpd.cs.princeton.edu/Papers/BarberDobkinHuhdanpaa.pdf [8]https://docs.scipy.org/doc/scipy/reference/generated/scipy.spatial.Delaunay.html [9]https://stackoverflow.com/questions/37363755/python-mouse-click-coordinates-as-simpl y-as-possible

[10][https://medium.com/@srosamazaid/the-greedy-algorithm-pattern-an-in-depth-analysis-7b b28d5dbfa7](https://medium.com/@srosamazaid/the-greedy-algorithm-pattern-an-in-depth-analysis-7b%20b28d5dbfa7)

**Додатки**

github: https://github.com/NeonKushnir/PrisonYardProblem