Науково-практичний звіт на тему

ЗАДАЧА ОРТОГОНАЛЬНОГО ТЮРЕМНОГО ДВОРУ

П.С. Пучек, 3 курс, група комп'ютерна математика 2

Анотація. У роботі розглянуто задачу ортогонального тюремного двору, яка є варіантом класичної музейної проблеми видимості, де необхідно розмістити камери для охорони як внутрішньої, так і зовнішньої частини ортогональних полігонів. Виконано побудову тріангуляції Делоне для визначення оптимальних камер та застосовано жадібний алгоритм для покриття всіх трикутників. У результаті досягнуто практичного рішення для задачі з обчислювальною складністю $O(n^2)$.

Abstract. This paper addresses the orthogonal prison yard problem, a variant of the classical art gallery visibility problem, where cameras must be placed to monitor both the internal and external areas of orthogonal polygons. A Delaunay triangulation was performed to identify optimal camera positions, and a greedy algorithm was applied to cover all triangles. The resulting solution achieves a practical approach with a computational complexity of $O(n^2)$.

Зміст

1. Вступ

Проблематика задачі. Задача тюремного двору є різновидом класичної музейної проблеми — однієї з фундаментальних задач видимості, де необхідно розмістити охоронців так, щоб вони контролювали всю внутрішню область багатокутника [1]. Проте у варіанті тюремного двору вимоги є значно суворішими. Охоронці повинні охоплювати не тільки внутрішність полігональної області, але й зовнішню зону, що безпосередньо прилягає до периметра [2]. Саме ця подвійна умова — одночасний контроль внутрішнього та зовнішнього середовищ — принципово відрізняє проблему тюремного двору від стандартної art-gallery problem та робить її складнішою з точки зору геометричних обмежень і алгоритмічної обробки.

Додаткові труднощі виникають у випадку ортогональних багатокутників. Хоча такі фігури задаються лише горизонтальними й вертикальними відрізками, вони створюють велику кількість рефлексних кутів і складні конфігурації видимості. Внаслідок цього навіть для полігонів з прямолінійною структурою побудова точних теоретичних меж або оптимальних алгоритмів розміщення охоронців залишається складною й недостатньо вивченою задачею.

Актуальність задачі. Задача ортогонального тюремного двору є актуальною як у теоретичних, так і в прикладних аспектах. Особливе значення мають ортогональні полігони, оскільки саме вони моделюють реальні просторові структури, такі як приміщення, архітектурні плани та технічні об’єкти [1]. Це робить задачу важливою для оптимального розміщення камер спостереження, сенсорів та інших систем моніторингу у сферах безпеки й інженерії [3].

Попри відносну простоту геометричної постановки, задача залишається відкритою: точний мінімум охоронців для ортогональних полігонів досі не встановлений, а відомі на сьогодні верхні межі не є остаточними [2]. Невирішеність проблеми, у поєднанні з її місцем у ширшому контексті задач видимості та їх алгоритмічної складності [4], підкреслює важливість подальших досліджень, спрямованих на уточнення меж, побудову ефективних алгоритмів та глибше розуміння структури ортогональних полігонів.

Аналіз існуючих підходів і методів розв’язання. Більшість класичних підходів до задач видимості в полігонах ґрунтується на комбінаторних методах та геометричних розбиттях. Основним інструментом виступає тріангуляція полігона, після чого аналіз зводиться до задач графів або розфарбувань, що дозволяє

отримувати верхні межі кількості необхідних охоронців [1]. Такі методи добре досліджені, але їх пряме застосування до задачі тюремного двору дає недостатньо точні результати через необхідність контролю зовнішнього простору.

Суттєвий прогрес у вивченні ортогональних полігонів було досягнуто у роботах Гоффмана та Кріґера [2], які показали, як розбивати такі полігони на прямолінійні структурні блоки з контрольованою видимістю. Цей підхід дає найкращі сучасні верхні межі та дозволяє краще зрозуміти зв’язок між структурою полігона й кількістю необхідних охоронців.

Паралельно розвивалися методи, орієнтовані на практичні застосування. Наприклад, опрацювання сенсорних мереж або оптимізація розміщення камер у реальних просторах використовує алгоритмічні наближення та жадібні стратегії, що демонструють свою ефективність у задачах з великою кількістю вузлів [3].

З точки зору теорії складності встановлено, що визначення мінімальної кількості охоронців навіть у базовому формулюванні є NP-важкою задачею [4]. Це означає, що точні оптимальні алгоритми для складніших варіантів, включно з задачею тюремного двору, є малоймовірними, і саме тому використовуються структурні розбиття та апроксимаційні стратегії.

2. Основні означення, теореми, леми, твердження, гіпотези

2.1. Означення

Означення 1 (ортогональний багатокутник). Багатокутник називається *ортогональним*, якщо кожне його ребро є або паралельним осі OX, або паралельним осі OY.

Такі багатокутники також називають *прямолінійними* або *rectilinear polygons* [6].

Означення 2 (галерея та охоронець). У класичній задачі галереї мистецтв охоронець — це точка всередині багатокутника, яка має лінії прямої видимості до певної частини площі багатокутника [1].

Означення 3 (точка-камера). У межах даного алгоритму камера — це обрана вершина, яка вважається такою, що покриває (освітлює) всі трикутники тріангуляції, які мають цю вершину спільною.

Означення 4 (тріангуляція). Тріангуляцією багатокутника називають розбиття його внутрішньої області на трикутники без накладань. Для множини точок часто використовується тріангуляція Делоне [1].

2.2. Теореми та леми

Теорема 1 (Art Gallery Theorem). Для будь-якого простого багатокутника з *n* вершинами достатньо $[\frac{n}{3}]$ охоронців. Доведення наведене у роботах Фіска та Хватала [5].

*Теорема 2 (варіант для ортогональних багатокутників)***.** Будь-який ортогональний багатокутник з *n* вершинами можна охороняти $[\frac{n}{4}]$ охоронцями.

Доведення цього факту наведено у роботах Кана, Кінга і Каца [6] та у препринті Гоффмана та Крігера [2].

Лема 1 (розбиття ортогонального багатокутника на прямокутники). Будь-який ортогональний багатокутник можна коректно розбити на прямокутні елементи шляхом проведення вертикальних і горизонтальних відрізків від рефлексних вершин [1].

2.3. Твердження та гіпотези алгоритму

Твердження 1 (про покриття). В рамках запропонованої моделі вважається, що якщо камера встановлена у вершині, вона повністю "бачить" (покриває) всі трикутники, що є суміжними з цією вершиною в графі тріангуляції.

Гіпотеза 1 (в контексті даного проекту). Оскільки знаходження точного мінімуму камер є NP-важкою задачею, а стандартна тріангуляція Делоне будується на опуклій оболонці точок, використання жадібного алгоритму («вибирай вершину, що належить найбільшій кількості непокритих трикутників») забезпечує субоптимальне, але достатнє для практичних цілей рішення за поліноміальний час.

3. Аргументація запропонованого підходу

Нехай $P \subset R^2$— простий ортогональний багатокутник з вершинами $V = \{v_1, \ldots, v_n\}$, де для кожного ребра (v_i, v_{i+1}) виконується рівність

$$(x_{i+1} - x_i)(y_{i+1} - y_i) = 0, \qquad (1)$$

що означає, що всі ребра є горизонтальними або вертикальними.

Проблема полягає у знаходженні множини вершин $S \subseteq V$ такої, що кожна точка $p \in P$ задовольняє умову видимості:

$$\exists s \in S: [p, s] \subseteq P,$$

де $[p, s]$ — відрізок між точками p та s. Це відповідає стандартній моделі "охоронця" (Vertex Guard) [1].
Оскільки пошук мінімальної множини таких точок для загальних багатокутників є $\exists$R-повною задачею [4], застосовується апроксимаційний підхід.
Розглянемо множину вершин V та її тріангуляцію T:

$$T = \{\Delta(v_i, v_j, v_k) \mid v_i, v_j, v_k \subseteq V\}.$$

Для будь-якої тріангуляції множина трикутників покриває полігон, тому справджується включення:

$$P \subseteq \bigcup_{\Delta \in T} \Delta.$$

Після побудови тріангуляції задача вибору камер зводиться до такої формальної моделі.
Для кожної вершини $v \in V$ розглянемо множину трикутників

$$C(v) = \{\Delta \in T: v \in \Delta\},$$

які містять цю вершину. Камера, розміщена у v, покриває всі трикутники з $C(v)$.
Потрібно знайти множину вершин $S \subseteq V$, таку, що кожен трикутник $\Delta \in T$ містить принаймні одну вершину зі S:

$$\forall \Delta \in T: \Delta \cap S \neq \emptyset.$$

Це математично еквівалентно задачі hitting set: множина S повинна мати хоча б один елемент у спільних вершинах кожного трикутника.

Жадібний алгоритм полягає у виборі вершини v^*, яка покриває найбільшу кількість трикутників, що ще не покриті, тобто:

$$v^* = arg\, max_{v \in V} |C(v) \cap T'|,$$

де T' — множина непокритих трикутників.

Після цього: $S := S \cup \{v^*\}, \quad T' := T' \backslash C(v^*)$.
Таке повторення гарантує повне покриття множини трикутників і, відповідно, всього полігону P, оскільки

$$P \subseteq \bigcup_{\Delta \in T} \Delta.$$

4. Опис алгоритму

Алгоритм розв’язує задачу визначення множини камер, розміщених у вершинах ортогонального багатокутника P, що забезпечують повне покриття його внутрішньої області. Формально багатокутник задано впорядкованою множиною

вершин $V = \{v_1, \ldots, v_n\}$, які утворюють простий замкнений контур, де всі ребра є горизонтальними або вертикальними.

Основна ідея алгоритму полягає у редукції задачі покриття полігону до задачі покриття множини трикутників. Для цього будується тріангуляція множини вершин багатокутника, після чого кожна вершина $v \in V$ розглядається як «кандидат на камеру», яка покриває всі трикутники, що її містять. Пошук камер зводиться до вибору множини вершин $S \subseteq V$, такої, що для кожного трикутника Δ з тріангуляції існує вершина $s \in S$, яка належить цьому трикутнику. Отже, покриття трикутників гарантує покриття всього полігону, оскільки полігон є об'єднанням трикутників.

Пошук найменшої множини S є NP-складною задачею покриття (set cover), тому застосовується жадібний алгоритм. На кожному кроці вибирається вершина, яка інцидентна найбільшій кількості трикутників, що ще не покриті. Після вибору такої вершини всі трикутники, що її містять, позначаються покритими та виключаються з подальшого розгляду. Ітерації продовжуються, доки не буде видалено всі трикутники тріангуляції.

Таким чином, алгоритм комбінує геометричну дискретизацію (тріангуляцію) та класичну жадібну стратегію покриття множин. Він гарантує коректність результату (повне покриття полігону) та має квадратичну обчислювальну складність, що є прийнятною для ортогональних багатокутників.

5. Етапи алгоритму

5.1 Перевірка коректності вхідних даних та ортогональності багатокутника

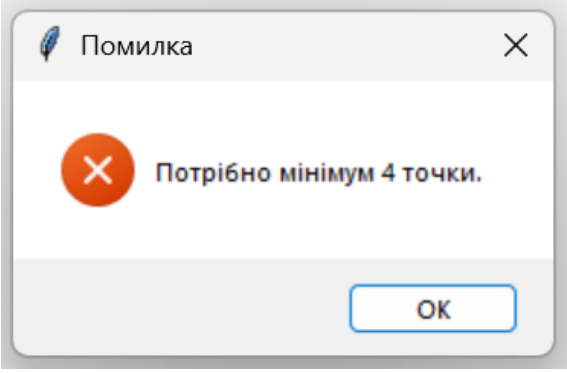
На початковому етапі алгоритм перевіряє, що кількість заданих вершин є достатньою для утворення простого багатокутника. Оскільки будь-який багатокутник повинен мати не менше ніж чотири вершини, виконується умова $n \geq 4$.

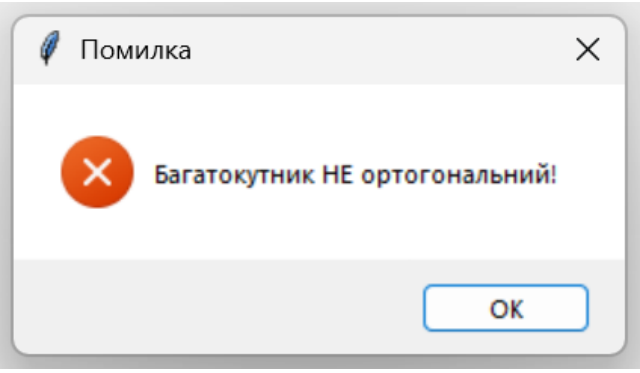
У випадку, коли множина точок не задовольняє цій умові, подальші етапи не виконуються відкривається відповідне вікно.

Після цього здійснюється перевірка ортогональності: для кожного ребра (v_i, v_{i+1}) перевіряється, що воно є або горизонтальним, або вертикальним.

Формально це вимагає виконання рівності (1). Якщо хоча б для одного ребра ця умова не виконується, багатокутник не є ортогональним, і алгоритм завершує роботу на цьому кроці і відкривається відповідне вікно.

Цей етап забезпечує коректність геометричної моделі, яка надалі використовується у тріангуляції та виборі камер.

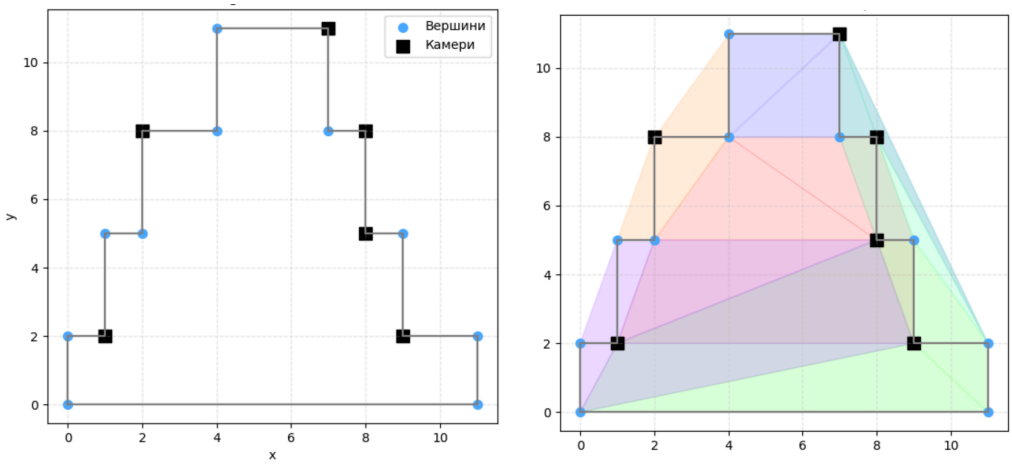




5.2 Побудова тріангуляції

Для множини точок на площині здійснюється побудова тріангуляції Делоне — структури, яка розбиває площину на неперетинні трикутники таким чином, щоб жодна точка не потрапляла всередину описаного кола будь-якого трикутника. Цей підхід забезпечує високі геометричні властивості: мінімальні кути трикутників є максимально великими, що гарантує стабільність і точність обчислень.
Процес побудови тріангуляції включає:

- Аналіз геометричного розташування точок: множина точок розглядається як дискретний набір вузлів у двовимірному просторі, на основі якого будується топологічна структура для визначення сусідніх точок.
- Формування комплексів трикутників: створення набору трикутників, що покривають всю опуклу оболонку множини точок, з використанням критерію порожнього кола.
- Упорядкування трикутників: тріангуляція дає набір трикутників, що утворюють планарний граф, де кожне ребро належить одному або двом трикутникам.



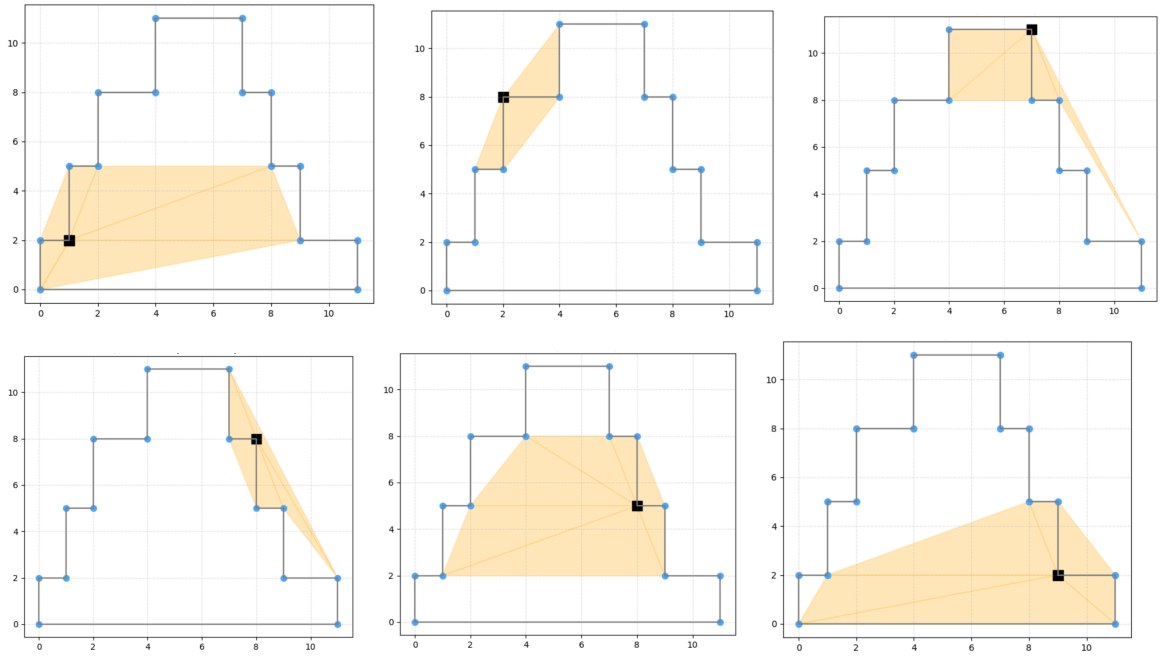
5.3 Жадібне покриття трикутників

Після побудови тріангуляції постає задача вибору підмножини вершин, які покривають усі трикутники. Для цього використовуємо жадібний алгоритм, що на кожному кроці вибирає найефективнішу точку, не виконуючи глобального пошуку. Кроки жадібного алгоритму:

1. Аналізуються всі вершини, і для кожної визначається кількість трикутників, в яких вона є частиною.
2. На кожній ітерації вибирається вершина, яка належить максимальній кількості ще непокритих трикутників. Це дозволяє максимізувати локальне покриття.
3. Після вибору вершини всі трикутники, що містять цю точку, видаляються з непокритих.
4. Процес повторюється, поки всі трикутники не будуть покриті.

5.4 Завершення та отримання результату

Після покриття всіх трикутників триангуляції формується множина вершин, що є кінцевим набором камер для спостереження. Кожна з цих камер покриває всі трикутники, забезпечуючи видимість внутрішньої частини багатокутника. Також можна переглядати зони видимості кожної камери.



6. Складність алгоритму

Нехай ортогональний багатокутник має n вершин.

- **Перевірка ортогональності** має складність $O(n)$, оскільки перевіряється кожне з nnn ребер на відповідність умовам горизонтальності або вертикальності.
- **Побудова тріангуляції Делоне** має складність $O(nlogn)$, оскільки для nnn точок тріангуляція Делоне виконується за цю кількість операцій.
- **Жадібне покриття трикутників** є найскладнішим етапом. Оскільки кількість трикутників ttt в Delaunay-тріангуляції складає $O(n)$, на кожній ітерації жадібного алгоритму обчислюється кількість трикутників, що містять кожну вершину, що дає складність $O(n^2)$.

Загальна складність алгоритму становить:

$$T(n) = O(n) + O(nlogn) + O(n^2) = O(n^2),$$

оскільки доданок $O(n^2)$ домінує. Квадратична складність узгоджується зі структурною складністю задачі охорони, яку Kahn–King–Katz [6] розглядають у контексті ортогональних багатокутників, та відповідає природній верхній межі для алгоритмів, що працюють із тріангуляція та покриттями.

7. Практична частина

7.1. Особливості програмної реалізації

Програмна реалізація цієї роботи охоплює кілька ключових етапів, зокрема перевірку ортогональності багатокутників, їх триангуляцію, а також визначення мінімальної кількості камер, необхідних для спостереження за всією територією багатокутника. Окрім того, реалізовано механізм інтерактивного малювання багатокутників за допомогою миші, а також вибору камер і відображення їх зон видимості.

Програма побудована на основі науково-обчислювальних бібліотек, таких як scipy, яка використовується для виконання триангуляції, matplotlib для візуалізації результатів, а також tkinter для створення графічного інтерфейсу користувача. Вона дозволяє працювати як з інтерактивними даними, так і з даними, що зчитуються з текстових файлів.

7.2. Опис основних функцій програмної реалізації

1. **Перевірка ортогональності багатокутника:** Функція is_orthogonal_polygon(points) виконує перевірку ортогональності багатокутника, визначаючи, чи є всі його ребра горизонтальними або вертикальними. Ортогональність є важливою умовою для коректної подальшої обробки багатокутника.
2. **Триангуляція багатокутника:** Функція delaunay_triangulation(points) використовує метод Делоне для триангуляції набору точок, що складають багатокутник. Триангуляція є необхідною для подальшого визначення камер спостереження, оскільки вона розбиває площу багатокутника на трикутники, які можна покривати камерами.
3. **Пошук камер для спостереження:** Функція find_cameras(triangles) реалізує жадібний алгоритм для визначення камер, що покривають всі трикутники, отримані під час триангуляції. Камери вибираються на основі найбільш часто зустрічаються точок, що належать кільком трикутникам.
4. **Візуалізація результатів:**
    - Функція draw_final відповідає за малювання багатокутника та розташування камер.
    - Функція draw_visibility візуалізує зони видимості камер для всіх трикутників.
    - Функція draw_visibility_single дозволяє відобразити зону видимості для конкретної камери.

5. **Графічний інтерфейс користувача:** Для зручності користувача було реалізовано графічний інтерфейс, що дозволяє:
    - Створювати багатокутники за допомогою миші (клас DrawingApp).
    - Завантажувати координати багатокутника з текстових файлів і здійснювати подальшу обробку (функція start_file_mode).
    - Обирати камери та переглядати їх зони видимості через окреме меню (select_camera_menu).

7.3. Характеристика вводу-виводу даних

1. **Вхідні дані:** Вхідними даними є координати точок, що складають багатокутник. Дані можуть бути введені безпосередньо через графічний інтерфейс (малювання мишкою) або завантажені з текстового файлу. У текстовому файлі кожен рядок містить пару чисел, що відображають одну точку багатокутника у вигляді "x, y".
2. **Вихідні дані:** Програма генерує візуалізацію багатокутника, розташування камер та їх зони видимості. Результати виводяться на екран у вигляді графічних вікон, створених за допомогою бібліотеки matplotlib.

7.4. Можливості, програмне та технічне забезпечення програмної реалізації

1. **Програмне забезпечення:**
    - Мова програмування: Python 3.x.
    - Бібліотеки:
        - scipy: для виконання тріангуляції і математичних обчислень.
        - matplotlib: для візуалізації геометричних фігур і результатів.
        - tkinter: для створення графічного інтерфейсу користувача.
2. **Технічні вимоги:**
    - Операційні системи: програму можна запускати на будь-якій платформі, що підтримує Python.
    - Наявність інтерпретатора Python та необхідних бібліотек.
3. **Можливості програми:**
    - Програма підтримує інтерактивне малювання багатокутників.
    - Виконується тріангуляція багатокутників та пошук мінімальної кількості камер для їх покриття.

- Візуалізація результатів з можливістю перегляду зон видимості кожної камери.
- Завантаження та обробка координат багатокутників з текстових файлів.

7.5 Лістинг основних модулів програми

1. **Модуль logic.py:**
   Модуль містить основну логіку перевірки ортогональності багатокутників, виконання тріангуляції та пошуку камер для покриття трикутників.
2. **Модуль draw.py:**
   Модуль відповідає за візуалізацію багатокутників, камер і їх зон видимості. Використовує бібліотеку matplotlib для побудови графіків.
3. **Модуль ui.py:**
   Модуль реалізує графічний інтерфейс користувача, що включає малювання багатокутників мишкою, завантаження даних з файлів, а також вибір і перегляд камер.
4. **Модуль main.py:**
   Модуль ініціює головне меню програми, в якому користувач може вибрати режим роботи програми (малювання багатокутників або завантаження з файлу).

8. Висновок

У ході виконання роботи було розглянуто алгоритмічні та програмні аспекти задачі ортогонального тюремного двору, що є складною задачею видимості для ортогональних полігонів. Основним результатом є реалізація алгоритму, що забезпечує мінімальне покриття трикутників у тріангуляції за допомогою жадібного підходу, що дає наближене, але практично ефективне рішення.

Проблеми, з якими стикнулися під час розробки, пов'язані з високою складністю точного визначення мінімальної кількості камер для ортогональних багатокутників. Оскільки задача є NP-важкою, жоден оптимальний алгоритм не може бути знайдений за поліноміальний час для великих вхідних даних. Тому було застосовано жадібне наближення, що забезпечує ефективність на практиці, але не гарантує точного мінімуму.

Загалом, хоча запропонований алгоритм є ефективним для багатьох практичних задач, є можливість для подальших вдосконалень, що можуть знизити обчислювальну складність і підвищити точність рішень.

9. Література

[1] J. O'Rourke. *Art Gallery Theorems and Algorithms*. Oxford University Press, New York, 1987, 31–80, 239–242.

[2] F. Hoffman, R. Krieger. *Guarding the Exterior and Interior of Orthogonal Polygons*. Manuscript, 2000, 2–6, 10–14.

[3] H. Kaul. *Art Gallery Problems*. Lecture Notes, Illinois Institute of Technology, 2020, 3–5, 18–21.

[4] M. Abrahamsen, K. Adiprasito, T. Miltzow. The Art Gallery Problem is ∃R-Complete. *Journal of the ACM*, 2021, 1–3, 5–7.

[5] S. Fisk. A short proof of Chvátal's art gallery theorem. *Discrete Mathematics*, 1978, 1–4.

[6] D. Kahn, J. King, M. Katz. *The art gallery problem for rectilinear polygons*. Computational Geometry, 2007, 10–18.