**Науково-практичний звіт на тему**

**Задача ортогонального тюремного двору**

Сокирко Н.Л., студентка 3 курсу, групи Комп’ютерна математика

**Анотація.** Задача мінімізації кількості відеокамер, розташованих на вершинах, щоб уся територія була видимою, на заданому багатокутнику (задача ортогонального тюремного двору). Обчислювальна складність задає два напрямки дослідження: розробка алгоритму, який знаходить приблизний розв’язок та визначення оптимального розв’язку для спеціальних класів простих багатокутників. У цій роботі я досліджую перший алгоритм – алгоритм апроксимації розв’язку, який опирається на загальний Генетичний метод розв’язку задачі тюремного двору.

**Abstract.** The problem of minimizing the number of guards placed on vertices needed to guard a given simple polygon (MINIMUM VERTEX GUARD problem) is NP-hard. This computational complexity opens two lines of investigation: the development of algorithms that determine approximate solutions and the determination of optimal solutions for special classes of simple polygons. In this paper we follow the first line of investigation proposing an approximation algorithm based on the general metaheuristic Genetic Algorithms to solve the MINIMUM VERTEX GUARD problem.

**1 Вступ**

*Постановка проблеми****.*** Задача тюремного двору є добре вивченою задачею видимості в обчислювальній геометрії. Першим задачу запропонував Віктор Клі, в 1973, коли він сформулював наступну проблему: Скільки стаціонарних вартових треба розставити, щоб покрити весь тюремний двір, у якого n стін? Неформально план тюремного двору змодельований за допомогою простих багатокутників (простий замкнений багатокутник з внутрішньою гранню) P та вартовими вважаються вершини P з 2π кутом видимості. Вважаємо, що точка x бачить точку y (або y видима для x) якщо відрізок, який сполучає х та у не перетинає зовнішнє шрань багатокутника P. Множина вартових покриває P, якщо кожна точка P є видимою для мінімум одного вартового. Таким чином, задача тюремного двору визначає мінімальну кількість вартових тюремного двору чий план є багатокутником, так що вони бачать всі вершини двору. Теорема тюремного двору: [n/3] вартових зрідка необхідно та завжди достатньо для забезпечення видимості багатокутника з n вершинами. Протягом років розглядалися та досліджувалися численні варіації початкової задачі. Цікавою задачею є задача Ортогонального тюремного двору. Теорема про ортогональний тюремний двір вперше була сформульована в 1983. Вона стверджує, що [n/4]вартових зрідка необхідно та завжди достатньо для забезпечення видимості багатокутника з n вершинами. Ортогональні прості багатокутники є важливим підкласом багатокутників. Справді, вони є корисними як апроксимація багатокутників; і вони з’являються природнім чином там, де домінують Декартові координати, наприклад, у растровій графіці, VLSI дизайні, або архітектурі. Ефективний алгоритм, який спирається на доведення наведеної теореми, був розроблений для встановлення видимості на ортогональних та неортогональних багатокутниках [n/4] та [n/3] вартовими, відповідно. Хоча в деяких випадках така кількість вартових необхідна, часто це набагато більше, ніж потрібно для видимості певного простого багатокутника. Наприклад, відомо, що для опуклих багатокутників потрібен лише один захисник. Варіантом цієї задачі є проблема Мінімальної кількості вартових (MVG), розміщених на вершинах (vertex-guard), необхідних для покриття даного простого багатокутника. Це NP-складна задача як для довільних, так і для ортогональних простих багатокутників.

*Аналіз останніх досліджень.*Готуючи дану лабораторну роботу, я знайшла статті, які пропонують розв’язки задачі тюремного двору.

1. У статті «Точний та ефективний алгоритм для ортогонального тюремного двору» авторів Marcelo C. Couto ; Cid C. de Souza ; Pedro J. de  
   Rezende запропоновано алгоритм, в якому охоронці можуть бути розташовані лише на вершинах багатокутника. Метод опирається на дискретизацію Р у скінченну множину точок у внутрішній частині багатокутника.
2. Стаття «Рій агентів для охорони тюремного двору: обчислювальне дослідження». Запропоновано два алгоритми з візуалізацією, де охоронці є автономними та мають брак комунікації. Для цього створюється симулятор, який здатен читати або генерувати не опуклі багатокутники та симулювати повороти охоронців всередині багатокутника, які зроблені за допомогою алгоритмів навігації.
3. Запропоноване рішення проблеми тюремного двору, яке є практичним ітеративним алгоритмом з охоронними точками, тому метод знаходить спадаючі верхні межі та зростаючі нижні межі допоки не буде досягнуто оптимальної величини.
4. Алгоритм апроксимації для мінімальної кількості охоронних вершин та ребер багатокутника з отворами або без них з загальною кількістю вершин n.

*Новизна та ідея.* У цій роботі я представляю експериментальні результати на простих багатокутниках. Алгоритм працює гарантовано ефективно як на ортогональних багатокутниках, так і на неортогональних.

*Мета статті.* Розробити узагальнений метод розв’язання задачі ортогонального тюремного двору.

**2 Основна частина.**

Сформулюємо геометричну постановку задачі ортогонального тюремного двору.

**Постановка задачі повного розділення множин.** Нехай P – множина вершин багатокутника. Задача полягає у знаходженні мінімальної кількості охоронців (відеокамер), які забезпечують видимість всієї внутрішньої частини багатокутника.

**2.1. Побудова розв’язку задачі розділення.** Спочатку ми будуємо багатокутник і використовуємо графічний інтерфейс для показу інструкції як можна користуватися програмою. У дизайні програми використано кнопки «Попередній» та «Наступний» для зручної навігації та візуалізації реалізації.

Я використала такі алгоритми:

* **Алгоритм «Ear Clipping» для тріангуляції**
* **М-розмальовка з використанням зворотного відстеження**
* **Видимість многокутника**

Розглянемо кожен з алгоритмів:

**Ear Clipping.**  Алгоритм тріангуляції складається з пошуку вуха і потім відрізання його від поточного многокутника. Початкова версія алгоритму відтинання вуха Мейстера виконується за час О(n3), з використанням часу О(n) для перевірки правильності побудованого трикутника.

**3-розмальовка зі зворотним відстеженням.** У цьому випадку дано неорієнтований граф. Використовується 3 кольори. Задача полягає у з’ясуванні, чи можливо призначити верщинам 3 різні кольори так, щоб не було двох інцидентних вершин одного кольору. Якщо розв’язок існує, алгоритм показує кольори, призначені вершинам.

Починаючи з вершини, де почалася побудова багатокутника, ми намагаємося призначити колір кожній вершині. Але перед призначенням перевіряємо, чи колір інцидентних вершин не є тим самим.

**Видимість многокутника.** У цій задачі дано простий багатокутник і ми рухаємося вздовж ребер з початкової вершини і якщо вона видимане має невидимих ребер з попередніми видимими вершинами, додаємо цю вершину до розв’язку. В іншому випадку переходить до найближчої точки перетину невидимого ребра і продовжує це робити, поки не пройде через весь багатокутник.

# Попередня обробка та структура даних. Дано простий багатокутник та вершина. Створюємо список для зберігання вершин. Якщо дані зчитуються з файлу, читаємо кожен рядок, який містить координату вершини, та додаємо його у список вершин багатокутника. Запускаємо алгоритм побудови багатокутника.

**Алгоритм.**

1. Дано початкову множину точок багатокутника V2. Створюємо початкових список R рефлексивних вершин та створюємо список Е для вух.
2. Починаємо з вершини створення багатокутника та вилучаємо одне вухо за один прохід, наприклад, V з Е. Додаємо трикутник Vi-1ViVi+1 до фінальної триангуляції. Видаляємо Vi зі списку V. Оновлюємо R та Е з інцидентними вершинами Vi-1, Vi+1. Якщо інцидентні вершини рефлексивні, тоді можливо, що вони стануть опуклими та можливе вухо. Якщо сусідня вершина є вухом, тоді дана вершина не обов’язково є вухом.
3. Повторити попередній крок до тих пір, поки V буде містити лише 3 вершини – останній трикутник тріангуляції.
4. Виконуємо 3-розмальовку вершин. Алгоритм описано вище.
5. Створюємо список невидимих вершин – пустий. Для кожної вершини багатокутника: якщо вершина є видимою та якщо немає невидимих ребер, які сполучають її з попередньою видимою вершиною, додаємо цю вершину до розв’язку.
6. Знаходимо найближчу точку перетину невидимих ребер з лінією (вершина, попередня видима вершина) та лінією (вершина, V) від вершини так, що V або попередня видима вершина належить сегменту (вершина, перетин).
7. Якщо два ребра, які виходять з попередньої видимої вершини знаходяться на одній стороні ліні (вершина, попередня видима вершина), та якщо лінія від найближчої точки перетину до лінії (вершина, попередня видима вершина) до попередньої видимої вершини багатокутника, тоді додаємо точку перетину до розв’язку.
8. Якщо два ребра сходяться в попередній видимій вершині і знаходяться на одній стороні лінії (вершина, попередня видима), якщо лінія від найближчої точки перетину до лінії (вершина, попередня видима) до попередньої видимої вершини многокутника, тоді додаємо вершину до розв’язку.
9. Якщо вершина невидима, додаємо її до списку невидимих вершин.

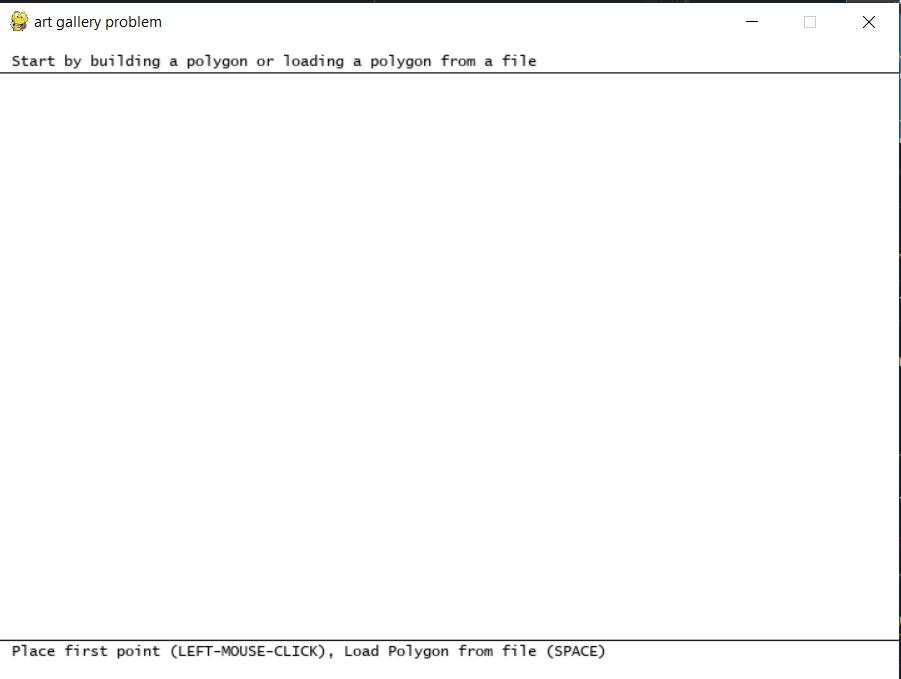
**3 Обґрунтування складності**

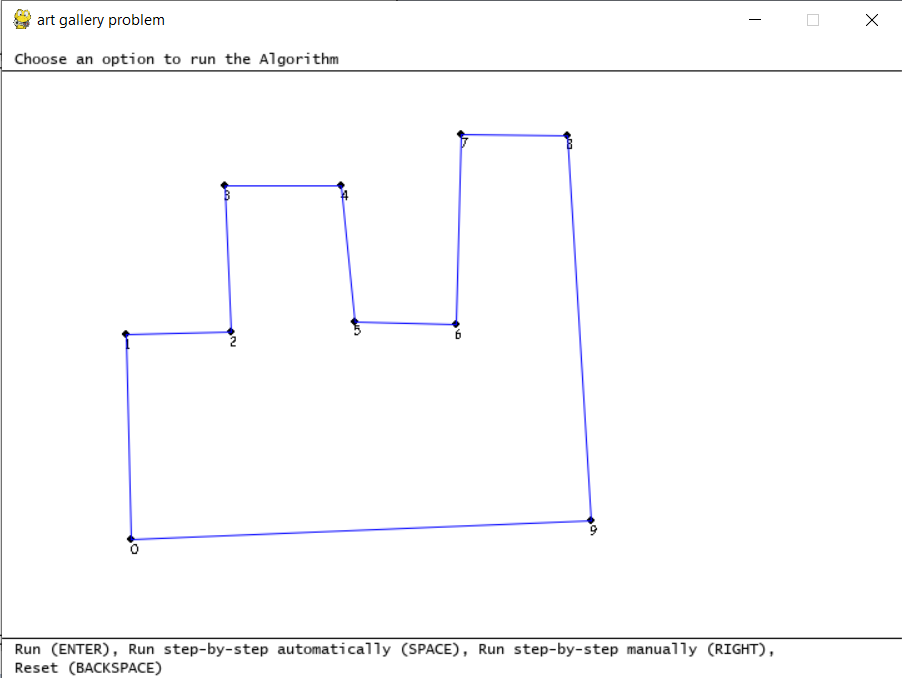
**Теорема 1.** Часова складність розв’язання задачі тюремного двору О(N).

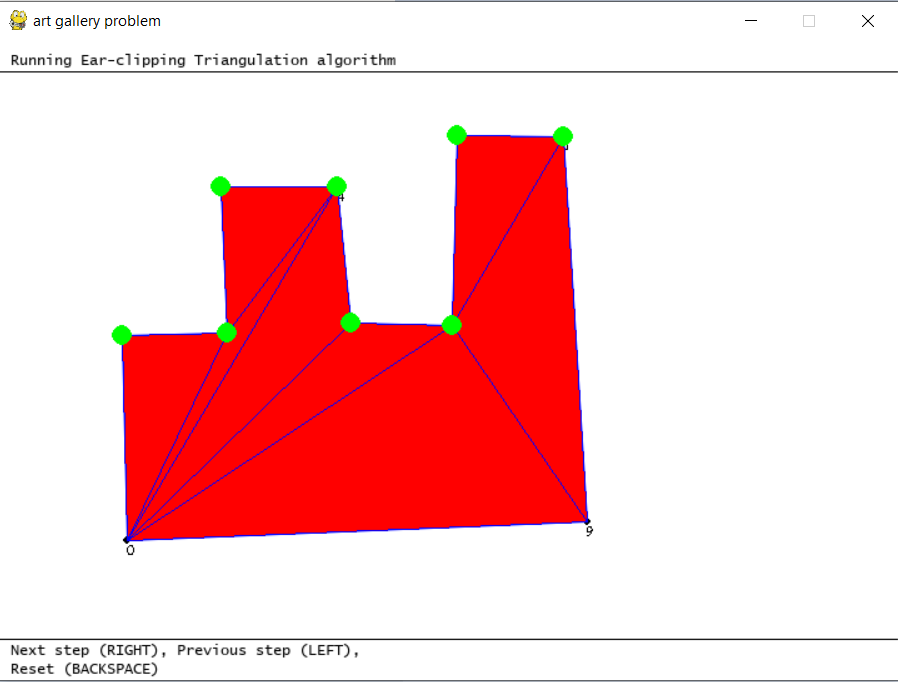
*Доведення.* Використано методи відтиння вух для тріангуляції, М-розмальовка та видимість багатокутника, які мають складність O(NlogN), O(N), O(N) відповідно, що в результаті дає загальний час O(N).

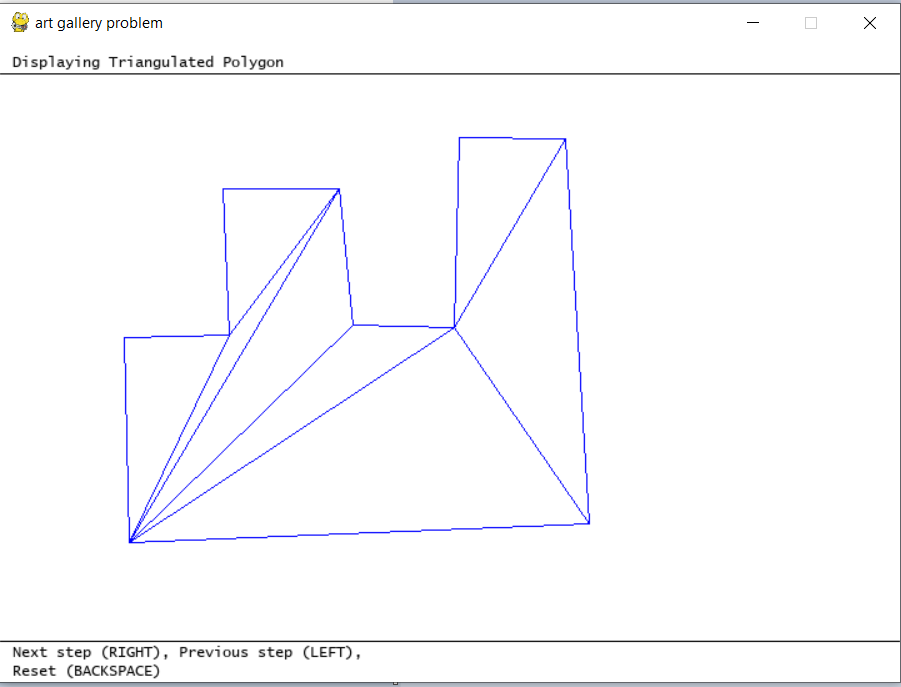
**4 Практична частина**

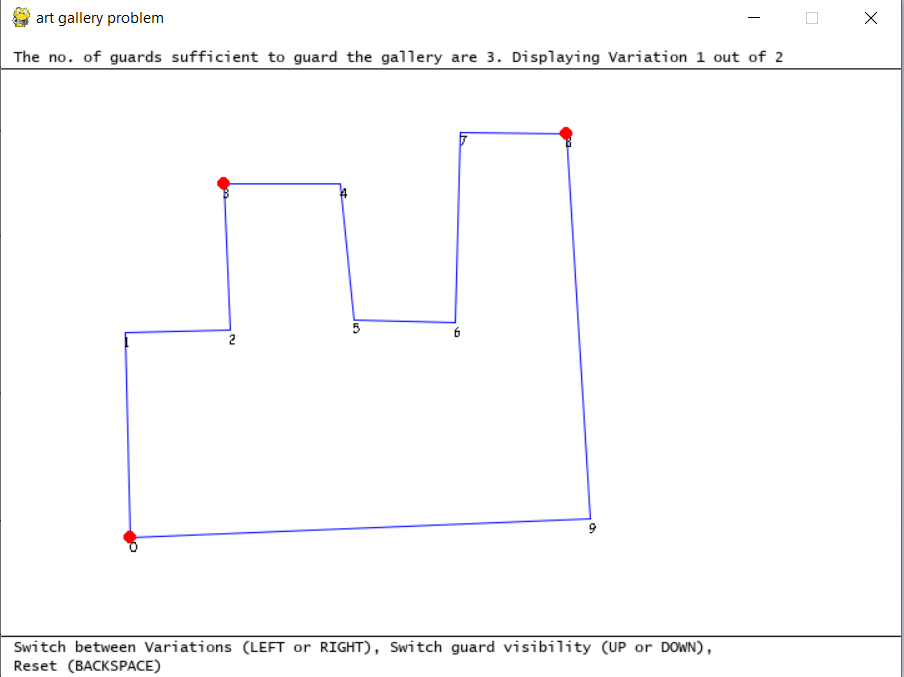
Особливістю даної реалізації є можливість роботи на багатокутниках будь-якого типу. Програма здатна працювати у двох режимах: зчитувати точки багатокутника, які поставлено курсором миші на робочій області, та зчитуванням даних з текстового файлу. Реалізовано зручний алгоритм навігації: після введення точок натиснути Enter – програма завершить побудову многокутника, Backspace – робоче поле очиститься, Space – запуск автоматичної презентації роботи програми step-by-step, стрілка вправо – наступний крок, стрілка вліво – назад. На останніх кроках програма показує всі можливі розв’язки та виводить на екран підсумок. Якщо написнути кнопку Esc, програма завершить свою роботу і закриється.











**5 Висновки**

Пошук точної кількості охоронних вершин, потрібних для видимості всього тюремного двору виконується за О(N). Щоб простий багатокутник був повністю видимий, потрібно n/3 вершнин-охоронців, для ортогонального випадку n/4. Видимість багатокутника є неточною у рідкісних випадках через порівняння плаваючої точки. Тріангуляція може помилитися інколи, якщо точки будуть розміщені занадто близько одна до одної. Програма не завжди дає мінімальну кількість охоронних вершин.

**Список літератури**

1. Amit, Y., Mitchell, J.S.B., Packer, E.: Locating Guards for Visibility Coverage of Polygons. In: ALENEX, pp. 120–134 (2007)
2. Bottino, A., Laurentini, A.: A nearly optimal sensor placement algorithm for boundary coverage. Pattern Recognition 41(11), 3343–3355 (2008)
3. Chvátal, V.: A Combinatorial Theorem in Plane Geometry. Journal of Combinatorial Theory (B) 18, 39–41 (1975)
4. Computational Geometry Algorithms Library, [http://www.cgal.org](http://www.cgal.org/)
5. Couto, M.C., de Rezende, P.J., de Souza, C.C.: An Exact and Efficient Algorithm for the Orthogonal Art Gallery Problem. In: SIBGRAPI 2007: Proceedings of the XX Brazilian Symposium on Computer Graphics and Image Processing, pp. 87–94. IEEE Computer Society, Washington, DC (2007)
6. Couto, M.C., de Souza, C.C., de Rezende, P.J.: Experimental Evaluation of an Exact Algorithm for the Orthogonal Art Gallery Problem. In: McGeoch, C.C. (ed.) WEA 2008. LNCS, vol. 5038, pp. 101–113. Springer, Heidelberg (2008)
7. Couto, M.C., de Rezende, P.J., de Souza, C.C.: An Exact Algorithm for an Art Gallery Problem. Technical report, Institute of Computing, University of Campinas (November 2009)
8. Fisk, S.: A Short Proof of Chvátal’s Watchman Theorem. Journal of Combinatorial Theory (B) 24, 374–375 (1978)
9. Lee, D.T., Lin, A.K.: Computational Complexity of art gallery problems. IEEE Transactions on Information Theory 32(2), 276–282 (1986)
10. Le Thi, H.A., Pham Dinh, T., Muu, L.D.: Exact penalty in DC programming. Vietnam Journal of Mathematics 27, 169–178 (1999)
11. Le Thi, H.A., Moeini, M., Pham Dinh, T.: Portfolio Selection under Downside Risk Measures and Cardinality Constraints based on DC Programming and DCA. Computational Management Science 6(4), 477–501 (2009)
12. Klawe, M., Kleitman, D.: Traditional art galleries require fewer watchmen. SIAM Journal on Algebraic and Discrete Methods 4(2), 194–206 (1983)

**Додатки**