УДК 519.68

Терещенко В.М., д. ф.-м.н., проф.
Корюкалов О.В., аспірант

Пошук найкоротшого шляху на множині геометричних об'єктів

Київський національний університет імені Тараса Шевченка, 83000, м. Київ, пр-т. Глушкова 4д,
e-mail: v_ter@ukr.net
e-mail: karia@bigmir.net

V.M. Tereshchenko, Dr. Sci. (Ph.-M.), Professor
O.V. Koriukalov, PhD student

Searching shortest paths on a set of geometric objects

Taras Shevchenko National University of Kyiv, 83000, Kyiv, Glushkova st., 4d,
e-mail: v_ter@ukr.net
e-mail: karia@bigmir.net

Робота присвячена дослідженню задачі пошуку найкоротшого шляху між двома заданими точками на множині геометричних об'єктів. В статті представлено огляд та аналіз основних алгоритмів.

Ключові слова: триангуляція, найкоротший шлях, багатокутна область.

The paper is devoted to research the problem of finding shortest path between two given points on given geometrical objects set. The article provides an overview and analysis of key algorithms. Computing a shortest path in a geometric domain have many applications in robotics, geographic information systems, wire routing, etc. The most basic form of the problem is: given a collection of obstacles, find a Euclidean shortest obstacle-avoiding path between two given points. A much broader collection of problems is defined by several parameters, that define the problem: objective function, constraints on the path, input geometry, dimension of the problem, type of moving object, single shot vs. repetitive mode queries, static vs. dynamic environment, exact vs. approximate algorithms, known vs. unknown map. In most cases open problems from this scope are improving performance, or reducing memory of existing algorithms. For some tasks (for example traveling salesman problem, computing the optimal path with obstacles in three-dimensional space), it is proved that they are NP-complex, and it is impossible to improve existing algorithms. Moreover, for other classes of problems, such as searching shortest path on polyhedral surface, we can improve the time complexity of algorithm by using a better data structures.

Key Words: triangulation, optimal path, polyhedral surface, polygonal domain.

Статтю представив д.ф.-м.н., проф. Анісімов А.В.

1. Вступ

Постановка проблеми. Обчислення оптимального шляху на множині геометричних обєктів — фундаментальна задача обчислювальної геометрії, що має широке застосування. Задача полягає у пошуці найкоротшого шляху між двома заданими точками, що оминає задані перешкоди. Така проблема часто виникає в робототехніці, плануванні руху, навігації, тощо.

Актуальність. В наш час алгоритми знаходження оптимальних шляхів за евклідовою мірою використовуються майже в усіх комп'ютерних іграх, геоінформаційних системах (GIS), при прокладанні дротів, автомобільних навігаторах та ін. Зв'язна міра використовується у робототехніці, коли зміна напрямку руху є набагато дорожчою за сам рух. Зв'язна міра без використання точок Штейнера використовується у задачах передавання прямолінійних бездротових сигналів, де необхідно мати можливість швидко визначати, як передати сигнал з однієї точки території до іншої з використанням найменшої кількості передавачів.

Метою даної статті є класифікація задач пошуку найкоротших шляхів, огляд базових алгоритмів, аналіз переваг та недоліків існуючих підходів.

2. Класифікація задач пошуку оптимального шляху

Сформулюємо загальну постановку задачі пошуку оптимального шляху на множині перешкод.

Загальна постановка задачі. *Дано множину k перешкод у евклідовому просторі (у вигляді геометричних фігур) між двома точками. Необхідно знайти найкоротший за евклідовою мірою шлях між двома точками, що оминає перешкоди.*

Систематизація задачі пошуку найкоротшого шляху є результатом декількох факторів, що визначають задачу: *цільова функція, додаткові обмеження до шляху, геометрія вхідних даних, розмірність, статика-динаміка, неповна інформація про геометрію перешкод, точність алгоритмічної реалізації, тощо.*

Далі коротко оглянемо існуючі класи задач, наведемо часові оцінки найкращих алгоритмів їх розв'язання, сформулюємо відкриті проблеми.

2.1. Цільова функція

У багатьох випадках при вирішені сформульованої вище задачі можуть постати наступні питання:

- яким чином ми вимірюємо довжину шляху?
- в якій метриці розглядається задача (евклідова метрика, L_1 метрика, L_p метрика, зв'язна міра і тд.)?

Зважаючи на це можна розглянути відповідні класи задач пошуку оптимального шляху на множині перешкод (ЗПОШМП) та відповідні алгоритми їх розв'язання.

Алгоритм пошуку оптимального шляху в багатокутній області за евклідовою метрикою O(nlogn), $O(n+h^2 logn)$. Прикладом алгоритму пошуку оптимального шляху в багатокутній області за евклідовою метрикою може бути наступний: спочатку побудуємо граф видимості [1] за час $O(E+nlogn)$, потім, використовуючи алгоритм Дейкстри, будуємо дерево найкоротших шляхів за $O(E+nlogn)$ [3]. Більш детально алгоритм описано в розділі 3.1.1.

Алгоритм пошуку оптимального шляху в багатокутній області за зв'язною мірою. В задачі пошуку мінімально-з'єднувального шляху, нашою метою є мінімізація кількості з'єднань (та відповідно кількості поворотів) в шляху, що сполучає *s* і *t*. У багатьох задачах, з'єднувальна відстань забезпечує більшу точність у вимірі складності шляху, ніж Евклідова довжина, так само як і при використанні програмного забезпечення для спрощення кривої. У роботі [2], було запропоновано алгоритм, що знаходить оптимальний за зв'язною мірою шлях за час $O(E_{VG}\alpha^2(n)logn)$, використовуючи області видимості.

Алгоритм пошуку оптимального шляху в багатокутній області за L_1 метрикою. Замість вимірювання довжини шляху відповідно за допомогою міри L_2 (Евклідової), розглядають задачу обчислення найкоротших шляхів в багатокутній області *P*, що є простішою для розв'язання за допомогою міри L_1[4]. Алгоритм пошуку оптимального шляху наступний: використовуючи неперервний метод Дейкстри, отримаємо граф найкоротших шляхів за $O(nlogn)$ часу, $O(n)$ пам'яті.

2.2. Додаткові обмеження до шляху

Інколи накладаються певні умови на характер шляху: пошук простого шляху від точки *s* до *t*, або пошук шляху з обов'язковим відвідуванням набору точок або областей і т.п.

Побудова мінімального зв'язувального дерева. Мінімальне кістякове дерево зв'язаного неорієнтованого графа — ациклічний зв'язний підграф цього графа, який містить всі його вершини, із найменшою можливою сумою ваг його ребер. Неформально кажучи, кістякове дерево складається з деякої підмножини ребер графа, таких, що рухаючись цими ребрами можна з будь-якої вершини графа потрапити до будь-якої іншої. Враховуючи той факт, що мінімальне зв'язувальне дерево є підграфом діаграми Делоне, в статті [5] наведено алгоритм, що обчислює його за час $O(nlogn)$.

Дерево Штейнера. Дерево, яке має найкоротшу довжину на множині *S* з *n* точок і задовольняє умові, що до множини *S* не заборонено додавати нові точки, називається деревом Штейнера. У статті [6] показано, що задача пошуку дерева Штейнера є NP-повною.

Задача комівояжера (TSP). Задача комівояжера полягає у знаходженні найвигіднішого маршруту, що проходить через вказані міста хоча б по одному разу. В умовах завдання вказуються критерій вигідності маршруту (найкоротший, найдешевший, сукупний критерій тощо) і відповідні матриці відстаней, вартості тощо. Зазвичай задано, що маршрут повинен проходити через кожне місто тільки один раз, в такому випадку розв'язок знаходиться серед гамільтонових циклів. Задача комівояжера для точок, що належать Евклідовій площині є NP-складною, що показано у статті [7].

2.3. Геометрія вхідних данних

Постановка задач ОШМП розрізняється також геометрією вхідних даних. Так, зокрема, геометричними фігурами множини вхідних даних можуть бути такі многокутники: опуклі, зіркові, прості, а також спеціальні многокутники (ізотетичні, з дірками) та фігури (кола, еліпси тощо).

Пошук оптимального шляху у простому многокутнику O(n). Основна задача геометричного пошуку найкоротшого шляху — знайти найкоротший шлях усередині *простого* багатокутника *P*, сполучаючи дві точки, *s* і *t*. Доповнення *P* служить перешкодою, яку шлях не може перетинати. Алгоритм пошуку оптимального шляху було запропоновано в статті [8]. Спочатку будується триангуляція простого багатокутника, дуальний граф якої буде деревом. Об'єднуючи трикутники вздовж рукавів, пошук найкоротшого шляху у дереві будується за $O(n)$. Більш детально алгоритм розглянуто в розділі 3.2. Пошук оптимального шляху у багатокутній області представлено розділі 3.1.1.

Тривимірний випадок задачі пошуку оптимального шляху на многранній поверхні подано в розділі 4.2. Залишається відкритм питання: чи можна обчислити найкоротший шлях на опуклому політопі за квадратичний час?

2.4. Тип рухомого об'єкту (автомату)

Поки що, ми розглянули лише задачу оптимально рухомого точкового автомату. Якщо автомат моделюється як круг, або як багатокутник, що не обертається, тоді більшість результатів просто переноситься із класичної задачі, застосовуючи стандартний підхід *конфігураційного простору* в плануванні руху: «скорочення» автомату до (необхідної) точки, і «зростання» перешкод (використовуючи суму Мінковського), таким чином, що доповнення збільшених перешкод моделює область площини. Оптимальний рух не круглих автоматів *що обертаються* — набагато складніша задача. Навіть найпростіший випадок руху (одиниці) лінійного сегменту в площині, надзвичайно нетривіальний.

2.5. Розмірність простору задачі

В залежності від розмірності простору в якому розглядається задача можна виділити такі задачі:

- задача пошуку оптимального шляху у багатокутній області на площині ($O(n\log n)$, $O(n+h^2\log n)$); детальна інформація міститься у розділі 3.1;

- задача пошуку оптимального шляху з перешкодами в тривимірному просторі $O(2^n)$.

У статті [11] було запропоновано алгоритм пошуку оптимального шляху в тривимірному просторі з перешкодами зі складністю $O(2^{2^{\wedge}n})$, що базується на зведенні оригінальної задачі до проблеми прийняття рішень на закритому полі дійсних чисел, який було оптимізовано до $O(2^n)$ у статті [10]. Детально алгоритм розглянуто в розділі 4.1.1.

Для цього випадку відкритими залишаються такі проблеми:

1. Чи може Евклідова задача пошуку найкоротшого шляху бути вирішена за час в $O(n+h\log n)$ області $O(n)$?
2. Використовуючи яку структуру даних можна ефективно обробити точний двоточковий запит в багатокутній області?
3. Як ефективно можна обчислити геодезичний центр/діаметр для багатокутної області?
4. Як ефективно можна розв'язати задачі Евкліда (пошуку найкоротших шляхів, геодезичного центру/діаметру) в моделі розрядної складності (замість реальної оперативної пам'яті) обчислення?

2.6. Точність алгоритмічної реалізації

В більшості випадків бажаний точний розв'язок задачі, проте не завжди просто його отримати. Наприклад, деякі задачі пошуку оптимального шляху зводяться до NP-повних, для яких існують наближені поліноміальні алгоритми їх розв'язання. Так у праці [12] запропоновано наближений поліноміальний алгоритм для цілого класу задач і в тому числі задачі побудови дерева Штейнера за час $O(n^{o(1/e)})$. Цей метод базується на оптимізації динамічним програмуванням певного класу мереж. Також він може бути використаний для задачі комівояжера.

Пошук оптимального шляху в тривимірному просторі з перешкодами. У роботі [13] було запропоновано наближений алгоритм який працює за час $O(n^2 \log n/e^4)$, що для знаходження оптимального шляху будує конусний граф видимості.

Пошук оптимального шляху на многогранній поверхні. У роботі [14] було запропоновано наближений алгоритм пошуку оптимального шляху на багатогранній поверхні за $O(n^{5/3} \log^{5/3} n)$, що базується на розбитті поверхні на $O(n/r)$ патчів, кожен з яких містить максимум r граней. Потім на границі кожного з

патчів обирається множина точок (порталів), що поєднуються у граф, до якого застосовуємо алгоритм Дейкстри.

2.7. Повнота інформації про геометрію перешкод

На практиці може постати задача, в якій не повна інформація щодо розміщення перешкод, наприклад, якщо в якості автомату виступає тактильний робот, що дізнається про структуру перешкод тільки коли дотикається до них, або робот, що дізнається про структуру перешкод коли вони потрапляють в зону його видимості.

Сконцентруємося на точних алгоритмах пошуку оптимального шляху для двовимірного та тривимірного випадків, а також на випадках із спрощеною геометрією вхідних даних.

3. Алгоритми пошуку оптимального шляху в двовимірному просторі

Розглянемо існуючі оптимальні алгоритми та їх модифікації.

3.1. Пошук оптимального шляху з перешкодами в багатокутній області

Алгоритм пошуку оптимального шляху в багатокутній області $O(n+h^2 \log n)$ **[19].** У статті [19] пропонується алгоритм, що обчислює найкоротший за евклідовою метрикою шлях в багатокутній області за час $O(n+h^2 \log n)$, використовуючи $O(n)$ пам'яті. Основна ідея алгоритму полягає у зведенні задачі до більш простої - пошуку оптимального шляху з перешкодами у вигляді опуклих багатокутників, що вирішується за час $O(n+h^2 \log n)$.

Спочатку триангулюємо багатокутну область за час $O(n+h \log h)$. Додамо s і t (початкова та кінцева точки шляху) до триангуляції, зв'язавши s (аналогічно t) з кутами трикутника якому вона належить. Позначимо через T отриману триангуляцію, двоїстий до T граф позначимо G_t. G_t є планарним графом з $O(n)$ вершинами, $O(n)$ ребрами і $h+1$ гранями.

Зменшимо граф G_t розділивши його на "коридори" наступним чином: спочатку вилучимо вершини з яких виходить лише одне ребро, потім вилучимо вершини яким інцидентні два ребра, замінивши відповідні два ребра на одне. Зрештою отримаємо граф (позначимо його G), що має тільки три-інциденті вершини, а саме $2h$-2 вершин, h+1 граней, $3h$-3 ребер (за формулою Ейлера). Кожній вершині G відповідає трикутник з T, назвемо такий трикутник вузловим. Границя коридору (рис. 1 [19]) складається з чотирьох частин: полігональний ланцюг вздовж перешкоди O_1 з a до b; діагональне ребро bc вузлового трикутника; полігональний ланцюг вздовж перешкоди O_2 з c до d; діагональне ребро da вузлового трикутника; Сегменти ad і bc є дверями коридору. Якщо знайти найкоротші шляхи з a до b і з c до d, отримаємо регіон в формі пісочного годинника H. На рис. 1 ліворуч цей регіон відкритий, праворуч - закритий. Позначимо через $Q < P$ регіон що містить усі вузлові трикутники, та "пісочні годинники". Очевидно, що для будь-яких s і t, оптимальний шлях буде належати Q.

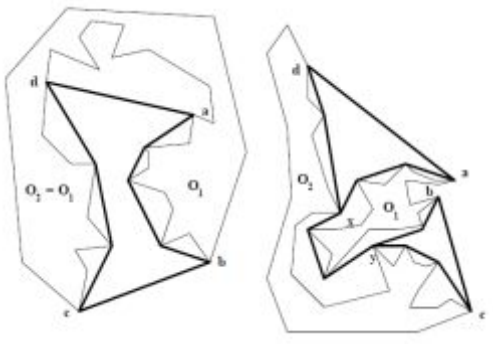

Рис. 1. Представлення коридору у вигляді двох полігональних ланцюгів та двох діагональних ребер.

Легко помітити, що границя Q (за виключенням ланцюгів закритих "пісочних годинників"), може бути представлена у вигляді $O(h)$ опуклих полігонів. Отже графі видимості Q буде містити $O(h^2)$ ребер.

Обчислимо складність алгоритму. Шляхи в усіх "коридорах" обчислюються за час $O(n)$. Використовуючи імплементацію алгоритму Дейкстри зі статті [1], пошук в графі видимості може бути здійснено за час $O(h^2 \log h)$. Крім того, на триангуляцію було витрачено $O(n + h \log h)$. Отже, сумарна часова оцінка складає $O(n + h^2 \log n)$.

Алгоритм пошуку оптимального шляху в багатокутній області **[23]** ***($O(n \log n)$, $O(n)$).*** В роботі [23] пропонується алгоритм пошуку оптимального шляху в багатокутній області з використанням $O(n \log n)$ часу та $O(n)$ памяті. Основна ідея алгоритму полягає в розбитті основної задачі на п'ять підзадач: регулярізація графа, виділення монотонних многокутників, триангуляція монотонних многокутників, локалізація точок, алгоритм A*. Коротко

опишемо та обгрунтуємо складність кожної підзадачі.

Регулярність графа розглядається відносно деякої прямої *l*, нехай це буде OX. Зорієнтуємо граф за зростанням *y* напрямку осі OX . Тоді вершину *v* будемо називати регулярною, якщо множини вхідних і вихідних з цієї вершини ребер одночасно не порожні. Граф називається регулярним відносно прямої (наприклад, осі OY або OX), якщо усі його вершини, окрім першої та останньої, регулярні. Для регуляризації графа використовується алгоритм плоского замітання [15] (*O(n log n)* - часу, *O(n)* - пам'яті).

Для виділення монотонних багатокутників будемо використовувати метод ланцюгів [16], що потребує *O(n log n)* часу.

Алгоритм триангуляції монотонних багатокутників використовує стек вершин, що розглядаються, та потребує *O(n)* часу.

Локалізацію точки в n-вершинному планарному підрозбитті можна реалізовувати методом ланцюгів [16] за час $O(\log^2 n)$ з використанням *O(n)* пам'яті, витрачаючи *O(n logn)* часу на передобробку.

Алгоритм A* [17] покроково розглядає усі шляхи, що ведуть з початкової вершини до кінцевої, доки не знайде мінімальний, і також в найгіршому випадку потребує *O(n logn)* часу.

Отже, загальна складність алгоритму буде дорівнювати сумі складностей його кроків, тобто *O(n logn)*.

3.2. Пошук оптимального шляху у простому многокутнику

Розглянемо існуючі оптимальні алгоритми та їх модифікації.

Алгоритм пошуку оптимального шляху з використанням триангуляції* [20-21] *за O(n). Беручи за основу статті B. Chazelle [20-21] задача пошуку найкоротшого шляху в простому багатокутнику може бути вирішена за час $O(n)$. Спочатку побудуємо триангуляцію багатокутника [20] за $O(n)$. Основна ідея запропонованого алгоритму полягає у створенні грубої тріангуляції на етапі знизу-вверх, і обчисленні результуючої триангуляції на етапі зверху-вниз, з використанням раніше обчисленої інформації по шляху. Основними методами, що використовуються є теорема про розріз полігона, яка дає нам схеми балансування, і теорема про лінійні сепаратори, що застосовується для побудови нових діагоналей. Побудувавши триангуляцію, розглянемо двоїстий до неї граф. Послідовність трикутників триангуляції що відповідають шляху з s до t назвемо “рукавом”. У статті [21] наведений алгоритм пошуку найкоротшого шляху в рукаві за час *O(n)*, використовуючи послідовне додавання трикутників триангуляції вздовж рукава.

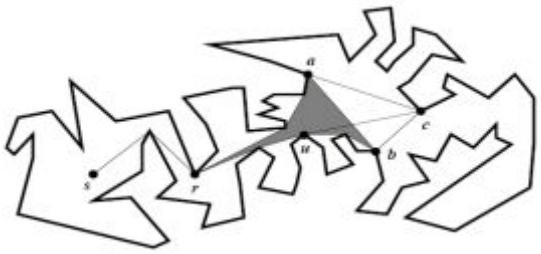

Рис. 2. Представлення рукава у вигляді воронки та двох полігональних ланцюгів.

“Рукав” складається з так званої “воронки” (з основою *ab*, та коренем *r*), та двох (увігнутих) найкоротших шляхів, що зєднують *r* з *a* і *b* відповідно, рис. 2 [18]. Додаючи до розгляду вершину *c*, ми розділяємо воронку на дві частини вздовж найкоротшого шляху з *r* до *c*, що буде містити відрізок *uc*, де *u* належить одному з увігнутих ланцюгів воронки. Після розділення зберігається властивість що новоутворена воронка (з основою *ac* або *bc*) також містить найкоротший шлях. Пошук точки *u* може бути здійснено за час *O(1)*, підтримуючи чергу вершин воронки. Триангуляцію простого багатокутника було виконано за час *O(n)*, пошук найкоротшого шляху з використанням “рукавів” також складає *O(n)*, отже, загальна часова оцінка складності алгоритму буде також *O(n)*, оскільки кожен з його кроків в найгіршому випадку потребує *O(n)* часу.

***Алгоритм пошуку оптимального шляху в простому багатокутнику з передобробкою за O(n), та часом запиту O(log n)* [22].** У статті [22] запропоновано алгоритм, що на етапі передобробки будує структуру даних за час *O(n)*, використовуючи яку, пошук найкоротшого шляху між двома точками може бути здійснено за *O(log n)*. Ідея, що лежить в основі методу не складна. Фаза попередньої обробки створює над основною тріангуляцією ієрархію вкладених субполігонів, таку, що будь-який найкоротший шлях перетинає тільки невелику кількість допоміжних субполігонів. Ми зберігаємо інформацію про найкоротші шляхи всередині кожного такого субполігону. Під час запиту, ми знаходимо найкоротший шлях між точками

запиту, агрегуючи інформацію обчислену для субполігонів.

Побудуємо двійкове дерево S декомпозиції вихідного полігона P. Коренем дерева буде власне діагональ *d*, що ділить P на два приблизно рівних субполігони P_1 і P_2, тоді діагоналі P_1 і P_2 будуть дітьми *d*, і т.д. рекурсивно для кожного P_i, поки P_i не буде трикутником. Найкоротші шляхи, що проходять через субполігон P_i перетинають дві діагоналі, які є границями P_i. Знайдемо фактор граф S*, додаючи до S ребра, що з'єднують *d* з батьківською вершиною, та усі діагоналі-границі P_d. Кожне ребро S* відображає пару діагоналей, для яких ми зберігаємо інформацію про найкоротший шлях, що їх перетинає. Отриманий фактор граф буде мати лінійну кількість вершин. Отже найкоротший шлях для будь-яких точок *s* і *t*, буде перетинати *O(log n)* діагоналей.

Алгоритм запиту (пошуку найкоротшого шляху між точками *s* і *t*) складається з чотирьох кроків: локалізація точок на триангуляції; знаходження підпослідовностей розділяючих діагоналей D_s і D_q; пошук "пісочного годинника" між *dp* і *dq*; об'єднання H(*p*, *dp*), H(*dq*, *q*) з H(*dp*, *dq*). Кожен з цих кроків має логарифмічну часову оцінку складності, отже загальна часова оцінка запиту складає *O(log n)*.

Алгоритм пошуку мінімального зв'язаного шляху у простому многокутнику (O(n*r²) передобробка, O(log r) запит) **[25].** У статті [25], представлено алгоритм, що на етапі передобробки будує структуру даних за час $O(n*r^2)$, використовуючи яку, пошук мінімального звязного шляху між двома точками може бути здійснено за *O(log r)*. Основна ідея алгоритму полягає в розбитті полігона на області видимості, в яких потім здійснюється локалізація точки.

Алгоритм попередньої обробки.

1. Спочатку будуємо граф видимості (*O(E)*).
2. Далі будуємо опуклу оболонку многокутника за алгоритмом D.T. Lee [26]. Опукла оболонка буде коренем дерева областей (*O(N)*).
3. Знайдемо список вікон, які утворюють усі вершини многокутника ($O(N^2)$) та додамо до цього списку початкові сторони многокутника. Усі відрізки списку будемо зберігати таким чином, щоб з лівого боку від відрізка була видима частина, а з правого боку – невидима. Нехай S– це кількість таких відрізків.
4. Далі у випадковому порядку додаємо вершини до дерева областей наступним чином: вносимо відрізок у вершину дерева областей; якщо вершина є листком, то розрізаємо відрізком на 2 області та записуємо частини що утворилися у відповідні нащадки цієї вершини, а відрізок – як відтинаючу пряму цієї вершини; інакше, перевіряємо позицію відрізка відносно відтинаючої прямої у вершині; якщо він не перетинає її – продовжуємо пошук у тому під-дереві, у яке потрапив відрізок; якщо перетинає – розділяємо відрізок на 2 частини та вставляємо ці частини у відповідні під-дерева; усі відрізки, що накладаються вилучаємо.

Нехай R – загальна кількість утворених областей. Для кожної з утворених областей видимості знаходимо центр та видимі точки з цього центру (*O(N)*). На основі цього будуємо дерево найкоротших шляхів пошуком у ширину (*O(E)*).

Алгоритм виконання запиту.

1. Перевіряємо точки на видимість, якщо прямої видимості немає – починаємо пошук.
2. Локалізуємо точки у дереві (*O(log r)*). Для цього ми у вершині дерева перевіряємо, з якого боку від відтинаючої прямої знаходиться точка і продовжуємо пошук у відповідному під-дереві, поки не дійдемо до листка. Якщо точка знаходиться на відтинаючій прямій то за нашою побудовою ліва частина для відрізка, який утворив відтинаючу пряму, буде видимою, отже пошук продовжувати будемо в лівому під-дереві.
3. Локалізувавши точки, знаходимо ту видиму вершину з точок видимості іншої області, до якої найкоротший шлях від першої області (*O(W)*), Відповідно ця довжина +2 буде мінімальною зв'язною довжиною, а сам шлях можна вивести за W, де W - його довжина (*O(W)*).

Обґрунтування складності. Для досягнення часу запиту *O(log N)* під час попередньої обробки знайдемо карту мінімальних шляхів між усіма областями видимості ($O(N*R^2)$ час, $O(R^2)$ пам'ять). Тоді під час пошуку необхідно буде лише локалізувати точки - *O(log R)*.

4. Алгоритми пошуку оптимального шляху в тривимірному просторі

4.1. Пошук оптимального шляху з перешкодами в тривимірному просторі

Алгоритм **[10]** ***($O(2^n)$ часу).*** У статті [10] пропонується алгоритм, що обчислює найкоротший за евклідовою метрикою шлях в тривимірному просторі з перешкодами за час

$O(2^n)$. Попередні найкращі результати мали часову оцінку складності $O(2^{2^{\wedge}n})$ і були запропоновані в статті [11]. Основна ідея алгоритму полягає у представленні вихідної задачі у вигляді виразу в теорії дійсно-замкненних полів. Нехай n - загальна кількість ребер перешкод; k - кількість "островів" (острів - опукла підмножина простору перешкод); $E = \{e_1, ..., e_n\}$ - множина ребер; якщо ребро $e = (u, v)$ має довжину l, тоді для $0<=d<=1$, $e(d)$ - точка на uv, що знаходиться на відстані dl від u, $e(0)=u$, $e(1) = v$; $s = e_1(0)$ - точка початок шляху, $t = e_n(0)$ - точка кінець шляху. Не порушуючи загальності можемо вважати що s і t кожна належить одному з "островів".

Відкрита формула $F(x_1 ... x_m)$ в теорії дійсно-замкненних полів складається з логічних операцій кон'юнкції, диз'юнкції, заперечення та нерівності на множині раціональних поліномів від змінних $x_1 ... x_m$. Формула з кванторами в цій теорії має вигляд $Q_1x_1 \dots Q_rx_r\ F(x_1 ... x_m)$, де Q_i - квантори, її ступінь - максимальна ступінь полінома в формулі F. Запропонований підхід полягає в тому, що для заданого l ми можемо описати формулою такого вигляду найкоротший шлях між s і t. Як показано у статті [3] дана формула в теорії дійсно-замкнених полів для довжини l, ступеню d, та кількості змінних v може бути перевірена на виконуваність за $dl^{O(v)}$. Ключовим моментом статті [10] є теорема, в якій показано що $v = O(\log k)$. Отже отримаємо одно-експотенційну загальну оцінку складності $n^{O(k)}$ роботи алгоритму.

4.2. Побудова оптимального шляху на многогранній поверхні

Алгоритм пошуку оптимального шляху на многогранній поверхні за $O(n^2)$ **[9].** У статті [9] було запропоновано алгоритм, що знаходить найкоротший шлях на необов'язково випуклій багатогранній поверхні за час $O(n^2)$. На відміну від використовуваних раніше підходів (J. S. B. Mitchell, D.M. Mount, C. H. Papadimitriou, $O(n^2 \log n)$ [27]), цей підхід відрізняється від неперервного алгоритму Дейкстри. Ключовим спостереженням є "один кут - одне розбиття", що задає верхню оцінку кількості гілок в дереві послідовностей. У статті спочатку розглядають випадок опуклого многогранника, який потім узагальнюють. Триангулюємо усі грані багатокутника S, також додамо до триангуляції вершини початку і кінця шляху s і t. Послідовність граней F задається як список суміжних граней таких що та містять спільне ребро . Будемо називати список послідовністю ребер. Використаємо метод, що називається площинна розгортка. Послідовність ребер буде розгорнута в такий спосіб: обертаємо навколо поки площини і не співпадуть і т.д. Будемо мати, що всі грані лежать в площині . Загальний вигляд найкоротшого локально шляху — шлях, який проходить через послідовність вершин та послідовність ребер, такий що розгортка шляху вздовж будь-якого ребра — це прямий відрізок і обидва кути шляху, що проходить через вершину більші або дорівнюють . Глобально найкоротший шлях геодезичний і має додаткову властивість: ніяке ребро не може з'явитися в більш ніж одній послідовності ребер і кожне ребро послідовності має бути простим.

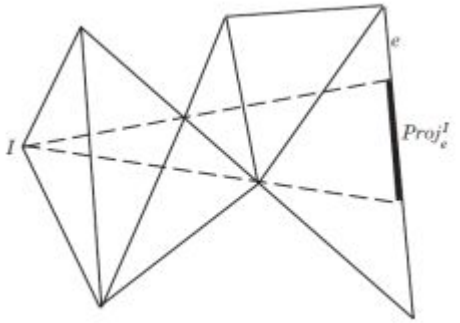

Рис. 3. Інтервальне вікно.

Тепер можемо побудувати дерево послідовностей, оскільки найкоротші шляхи з вершини s, можуть бути впорядковані за кутом під яким вони виходять з s. При побудові шляху на розгортці будемо використовувати проекції ребер (інтервальні вікна) – рис. 3 [9]. Дерево послідовностей, що містить послідовність проекцій ребер, буде експоненціального розміру, і матиме не більше n рівнів, де n — кількість граней. СН-агоритм уникає експоненційної складності, завдяки властивості "один кут — одне розбиття". Нижче наведений алгоритм побудови дерева послідовностей.

Алгоритм Чена [9].

1.Проводимо ініціалізацію: для всіх вершин, крім початкової, встановлюємо відстані в $+\infty$, потім асоціюємо кожний кут з нульовим інтервальним вікном та кожну вершину з нульовим псевдо-початковим вікном. Нарешті, вводимо FIFO чергу Q для зберігання подій.

2.Присвоюємо початковій вершині відстань 0, створюємо псевдо-початкове вікно *w* для *s* і кладемо *w* в *Q*.
3. ПОКИ *Q* не порожня і рівень не перевищує число граней n: беремо вікно *w* з *Q;* якщо *w* — псевдо-початкове, *w*=(*d, v*); якщо *d* < поточна оцінка відстані в *v;* оновлюємо значення відстані *v*; якщо *v* — сідлова - видаляємо всі старі псевдо-початкові вікна в *v* та його піддеревах.
4. Для кожного ребра, протилежного v, додаємо дочірне вікно (*d, v, e,* [*0, 1*]) у хвіст списка *Q*.
5. Оновлюємо відстань кожної вершини *v*' суміжної до *v* через *w* і додаємо псевдо-початкове вікно якщо в Q ІНАКШЕ // *w* — інтервальне, *w* = (*d, I, e,* [*a, b*]).
6. Якщо *w* має тільки одне дочірне вікно на лівому (правому) ребрі або *w* не зайняло протилежний кут у *w*', тоді: рахуємо лише дочірні і заносимо в *Q*; ІНАКШЕ // *w* займає протилежний кут у *w.*
7. Видалити скасоване піддерево *w*'.
8. Порахувати два дочірні дерева *w* і занести в *Q*.
9. Перевірити чи *w* може давати найкоротшу відстань до *v*, протилежно ребру *e*; якщо так, оновлюємо відстань у *v*; і якщо *v* — сідлова або гранична, необхідно також згенерувати псевдо-початкове вікно в *v* і вставити в чергу *Q*.

Фактично ми побудували розріз многогранника, за найкоротшими шляхами до кожної з його вершин. J. Chen та Y. Han довели, що загальне число згенерованих вузлів (включаючи скасовані) рівне $O(n^2)$. Отже загальна часова оцінка складності $O(n^2)$.

У статті [24] пропонується два способи покращення алгоритму Чена для пошуку мінімального шляху на многогранній поверхні. Основна ідея полягає в тому, щоб відсіяти непродуктивні вікна, які значно погіршують час роботи алгоритму в загальному випадку і витрачають багато пам'яті. Інша ідея полягає в належній підтримці черги з пріоритетами, так як і в алгоритмі Дейкстри.

5. Висновки

В даній роботі представлено огляд існуючих задач пошуку оптимального шляху, та проведено аналіз базових алгоритмів їх розв'язання. Систематизація задачі пошуку найкоротшого шляху є результатом декількох факторів, що визначають задачу: міра довжини шляху, геометрія множини об'єктів, на яких здійснюється пошук, вимір простору, і тд. Відкриті проблеми даного напряму в більшості випадків полягають в покращенні швидкості роботи, або зменшені пам'яті існуючих алгоритмів. Для деяких задач (наприклад, задача комівояжера, пошук оптимального шляху з перешкодами в тривимірному просторі), доведено що вони є NP-складними, тобто суттєво покращити існуючі алгоритми неможливо. При цьому для інших класів задач, наприклад пошук оптимального шляху на многогранній поверхні, можливе покращення часової оцінки складності алгоритму за рахунок використання більш вдалої структури даних, що і планується втілити в наступних роботах.

Список використаних джерел

1. *Riviere S.* Topologically sweeping the visibility complex of polygonal scenes / S. Riviere. - In Proc. 11th Annu. ACM Sympos. Comput. Geom. - 1995. - P. 36-37.
2. *Maheshwari A.* Link distance problems / A. Maheshwari, J.-R. Sack, J. Urrutia. - Handbook of Computational Geometry // Elsevier Science Publishers B.V. North-Holland, Amsterdam. - 1998. - P. 112-116.
3. *Fredman M.* Fibonacci heaps and their uses in improved network optimization problems / M. Fredman, R. E. Tarjan. - J. ACM — 1987. - P. 596-615.
4. *Mitchell J. S. B.* L1 shortest paths among polygonal obstacles in the plane / J. S. B. Mitchell. - Algorithmica. - 1992. - P. 55-88.
5. *Cheriton D.* Finding minimum spanning trees / D. Cheriton, R. E. Tarjan. - SIAM J. Comput. - 1976. - P. 724-742.
6. *Garey M. R.* The complexity of computing Steiner minimal trees / M. R. Garey, R. L. Graham, D. S. Johnson. - SIAM J. Appl. Math. - 1977. - P. 835-859.
7. *Papadimitriou C. H.* The Euclidean traveling salesman problem is NP-complete / C. H. Papadimitriou. - Theoret. Comput. Sci. - 1977. - P. 237-244.
8. *Lee D. T.* Euclidean shortest paths in the presence of rectilinear barriers / D. T. Lee, F. P. Preparata. - Networks 14. - 1984. - P. 393-410.
9. *Chen J.* Shortest paths on a polyhedron / J. Chen, Y. Han. - In Proc. 6th Annu. ACM Sympos. Comput.Geom. - 1990. - P. 360-369.

10. *Reif J. H.* A single-exponential upper bound for finding shortest paths in three dimensions / J. H. Reif, J. A. Storer. - J. ACM – 1994. - P. 1013-1019.
11. *Sharir M.* On shortest paths in polyhedral spaces / M. Sharir, A. Schorr. - SIAM J. Comput. - 1986. - P. 193-215.
12. *Mitchell J. S. B.* Guillotine subdivisions approximate polygonal subdivisions: A simple new method for the geometric k-MST problem / J. S. B. Mitchell. - In Proc. 7th ACM-SIAM Sympos // Discrete Algorithms – 1996. - P. 402-408.
13. *Clarkson K. L.* Approximation algorithms for shortest path motion planning / K. L. Clarkson. - In Proc. 19th Annu. ACM Sympos. Theory Comput. - 1986. - P. 56-65.
14. *Varadarajan K. R.* Approximating shortest paths on an nonconvex polyhedron / K. R. Varadarajan, P. Agarwal. - In Proc. 38th Annu. IEEE Sympos. Found. Comput. Sci. - 1997. - P. 132.
15. *Препарата Ф.* Вычислительная геометрия: Введение / Ф. Препарата, М. Шеймос. - [пер. с англ.] – 1989. - P. 117.
16. *Edelsbrunner E.* Optimal point location in a monotone subdivision / E. Edelsbrunner, L.J. Guibas, J. Stolfi. - SIAM J. Computing. – 1986. – № 15(2). – P. 317-340.
17. *Lester P.* A* Pathfinding for Beginners / P. Lester. – 21 Oct. 2006 [Електронний ресурс]. – Режим доступу : http://www.policyalmanac.org/games/aStarTutorial.htm
18. *Mitchell J. S. B.* Geometric Shortest Paths and Network Optimization / J. S. B. Mitchell. – 1998. - P. 320-334.
19. *Kapoor S.* An efficient algorithm for Euclidean shortest paths among polygonal obstacles in the plane / S. Kapoor, S. N. Maheshwari, J. S. B. Mitchell. - Discrete Comput. Geom. - 1997. - P. 377-383.
20. *Chazelle B.* A theorem on polygon cutting with applications / B. Chazelle. - In Proc. 23rd Annu. IEEE Sympos. Found. Comput. Sci. - 1982. - P. 339-349.
21. *Chazelle B.* Triangulating a simple polygon in linear time / B. Chazelle. - Discrete Comput. Geom. - 1991. - P. 485-524.
22. *Guibas L. J.* Optimal shortest path queries in a simple polygon / L. J. Guibas, J. Hershberger. - J. Comput. Syst. Sci. - 1989. - P. 126-152.
23. *Терещенко В.Н.* Подход к поиску оптимального пути между двумя точками на множестве преград / В. Н. Терещенко, Д. Янчик, Д. Пустовойтов, Е. Чернышов. - Штучний інтелект — 2010. - P. 113-119.
24. *Терещенко В. М.* Побудова мінімального шляху на многогранній поверхні в R3 / В. М. Терещенко. - Вісник Київського національного університету імені Тараса Шевченка — 2012. - P. 57-64.
25. *Терещенко В. М.* Пошук мінімального зв'язаного шляху у простому многокутнику / В. М. Терещенко, А. С. Трегубенко. - Вісник Київського національного університету імені Тараса Шевченка – 2012. - P. 95-99.
26. *Lee D. T.* On finding the convex hull of a simple polygon / D. T. Lee. - IJCIS, Vol.12, No. 2 -1983. — P. 87-98.
27. *Mitchell J. S. B.* The discrete geodesic problem / J. S. B. Mitchell, D. M. Mount, C. H. Papadimitriou. - SIAM J. Computing – 1987. P. 1260-1264.

References

1. RIVIERE S. (1995) *Topologically sweeping the visibility complex of polygonal scenes.* In Proc. 11th Annu. ACM Sympos. Comput. Geom, p. 36-37.
2. MAHESHWARI A., SACK J-R. (1998) *Link distance problems.* J.-R. Sack and J. Urrutia, editors, Handbook of Computational Geometry, Elsevier Science Publishers B.V. North-Holland, Amsterdam, p.39.
3. FREDMAN M., TARJAN R. E. (1987) *Fibonacci heaps and their uses in improved network optimization problems.* J. ACM, p. 34:596-615.
4. MITCHELL J. S. B. (1992) *L1 shortest paths among polygonal obstacles in the plane.* Algorithmica, p. 8:55-88.
5. CHERITON D., TARJAN R. E. (1976) *Finding minimum spanning trees.* SIAM J. Comput., p. 5:724-742.
6. GAREY M. R., GRAHAM R. L., JOHNSON D. S. (1977) *The complexity of computing Steiner minimal trees.* SIAM J. Appl. Math., p. 32:835-859.
7. PAPADIMITRIOU C. H. (1977) *The Euclidean traveling salesman problem is NP-complete.* Theoret. Comput., p. 4:237-244.
8. LEE D. T., PREPARATA F. P. (1984) *Euclidean shortest paths in the presence of*

rectilinear barriers. Networks 14, p. 14:393-410.

9. CHEN J., HAN Y. (1990) *Shortest paths on a polyhedron.* In Proc. 6th Annu. ACM Sympos. Comput.Geom., p. 360-369.

10. REIF J. H., STORER J. A. (1994) *A single-exponential upper bound for finding shortest paths in three dimensions.* J. ACM p. 41(5):1013-1019.

11. SHARIR M., SCHORR A. (1986) *On shortest paths in polyhedral spaces.* SIAM J. Comput. p. 15:193-215.

12. MITCHELL J. S. B. (1996) *Guillotine subdivisions approximate polygonal subdivisions: A simple new method for the geometric k-MST problem.* In Proc. 7th ACM-SIAM Sympos Discrete Algorithms, p. 402-408.

13. CLARKSON K. L. (1986) *Approximation algorithms for shortest path motion planning.* In Proc. 19th Annu. ACM Sympos. Theory Comput., p. 56-65.

14. VARADARAJAN K. R., AGARWAL P. (1997) *Approximating shortest paths on an nonconvex polyhedron.* In Proc. 38th Annu. IEEE Sympos. Found. Comput. Sci., p. 132.

15. PREPARTA F. (1989) *Vychyslitelnaia geometriia: Vvedenie.* Preparta F., Sheymos M., p. 117.

16. EDELSBRUNNER E. (1986) *Optimal point location in a monotone subdivision.* E. Edelsbrunner, L.J. Guibas, J. Stolfi, SIAM J. Computing, № 15(2), p. 317-340.

17. LESTER PATRICK. (21 Oct. 2006) *A* Pathfinding for Beginners.* Available from: http://www.policyalmanac.org/games/aStarTutorial.htm

18. MITCHELL J. S. B. (1998) *Geometric Shortest Paths and Network Optimization*. p. 320-334.

19. KAPOOR S., MAHESHWARI S. N., MITCHELL J. S. B. (1997) *An efficient algorithm for Euclidean shortest paths among polygonal obstacles in the plane.* Discrete Comput. Geom., p. 18:377-383.

20. CHAZELLE B. (1982) *A theorem on polygon cutting with applications.* In Proc. 23rd Annu. IEEE Sympos. Found. Comput. Sci., p. 339-349.

21. CHAZELLE B. (1991) *Triangulating a simple polygon in linear time.* Discrete Comput. Geom., p. 6:485-524.

22. GUIBAS L. J. HERSHBERGER J. (1989) *Optimal shortest path queries in a simple polygon.* J. Comput. Syst. Sci. - P. 39:126-152.

23. TERESHCHENKO V.N., YANCHIK D., PUSTOVOYTOV D., CHERNISHOV E. (2010) *An Approach to finding the optimal path between two points on a set of obstacles.* Shtuchnyi intelekt, p. 113-119.

24. TERESHCHENKO V.N. (2012) *Constructing minimum path on the polyhedral surface in R3.* Taras Shevchenko National University of Kyiv's visnyk, p. 57-64.

25. TERESHCHENKO V.N., TREGUBENKO A.S. (2012) *Finding a minimum link path within a simplepolygon.* Taras Shevchenko National University of Kyiv's visnyk, p. 95-99.

26. LEE D. T. (1983) *On finding the convex hull of a simple polygon.* IJCIS, Vol.12, No. 2, p. 87-98.

27. MITCHELL J.S.B. MOUNT D. M. PAPADIMITRIOU C. H. (1987) *The discrete geodesic problem.* SIAM J. Computing, p. 1260-1264.

Надійшла до редколегії 17.09.15