УДК 004.42

***Жнакін Володимир Володимирович**, здобувач вищої освіти*
КПІ ім. Ігоря Сікорського, Україна
***Науковий керівник: Жаріков Едуард В'ячеславович**,*
доктор технічних наук, професор, завідувач кафедри інформатики та програмної інженерії, Національний технічний університет України "Київський політехнічний інститут імені Ігоря Сікорського".

РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ НА ОСНОВІ ПОКРАЩЕНОГО ГЕОМЕТРИЧНОГО АЛГОРИТМУ ДЛЯ ВИРІШЕННЯ ЗАДАЧІ ОПТИМІЗАЦІЇ ТРАНСПОРТНОГО РУХУ

Анотація. Сучасні системи керування транспортними засобами логістичних компаній потребують розробки програмного забезпечення, здатного прораховувати найкоротші відстані з невеликою похибкою і за відносно короткий час, який враховуватиме те, що ці місця з'єднуються не прямою лінією, а складнішою фігурою. Основним недоліком більшості існуючих реалізацій є те, що вони не аналізують існуючі дороги між місцями, а обчислюють лише найкоротші відстані по координатах, що призводить до неточних рішень, які неможливо застосувати у реальному світі. У цій роботі наведена постановка завдання пошуку найменшого можливого циклічного маршруту, який проходить через заданий набір міст, починаючи і закінчуючи в тому самому місті. У результаті, алгоритм повинен знайти послідовність відвідування міст, щоб загальна довжина шляху між ними була мінімальною, і шлях проходив через кожне місто рівно один раз. Проведено експериментальні дослідження з існуючими дорогами для 10. Результати для 10 міст порівняні з алгоритмом повного перебору, який також аналізує дороги. Наведено приклад реалізації покращеного геометричного алгоритму з використанням існуючих доріг мовою програмування swift для використання на платформі iOS.
КЛЮЧОВІ СЛОВА: ЗАДАЧА КОМІВОЯЖЕРА, ОПТИМАЛЬНИЙ ШЛЯХ, КОМБІНАТОРНА ОПТИМІЗАЦІЯ, MAPKIT, IOS.

Abstract. Modern vehicle management systems of logistics companies require the development of software capable of calculating the shortest distances with a small error and in a relatively short time, which will take into account the fact that these places are not connected by a straight line, but by a more complex shape. The main drawback of most existing implementations is that they do not analyze the existing roads between places, but only calculate the shortest distances by coordinates, which leads to imprecise solutions that cannot be applied in the real world. This paper presents the formulation of the task of finding the smallest possible circular route that passes through a given set of cities, starting and ending in the same city. As a result, the algorithm must find a sequence of visiting cities so that the total length of the path between them is minimal, and the path passes through each city exactly once. Experimental studies were conducted with existing roads for 10 cities. Results for 10 cities are compared with a full sweep algorithm that also analyzes roads. An example of the implementation of an improved geometric algorithm using existing roads is given in the swift programming language for use on the iOS platform.
KEY WORDS: TRAVELING SALESMAN PROBLEM, OPTIMAL WAY, COMBINATORY OPTIMIZATION, MAPKIT, IOS.

Вступ. У світі практично немає двох місць, які пов'язані один з одним найкоротшим відстанню – прямою лінією. Літаки літають з одного аеропорту в інший так, щоб якнайбільше часу летіти над сушею і пролітати якомога більше аеропортів [1]; кораблі пливуть з одного порту в інший якомога ближче до берегової лінії [2]; дороги для автомобілів будуються так, щоб вони були прокладені через найбільшу кількість населених пунктів – щоб якнайбільше людей могло нею користуватися [3]. У всіх цих випадках маршрутами транспортних засобів будуть далеко не прямі лінії, а дуги, ламані лінії чи складніші фігури. Для точніших розрахунків найкоротших відстаней між містами необхідно модернізувати реалізований геометричний алгоритм так, щоб він взаємодіяв як з координатами обраних міст, так і аналізував існуючі дороги, їх маршрути і відстані.

Основна частина. Для того, щоб геометричний алгоритм можна було використовувати з реальними даними, існуючими картами та дорогами, необхідно інтегрувати у додаток алгоритм, який після введення точок (міст), крім їх координат, також аналізуватиме існуючі маршрути, які є між цими точками (містами). На рисунку 1 продемонстровано як зараз працює геометричний алгоритм.

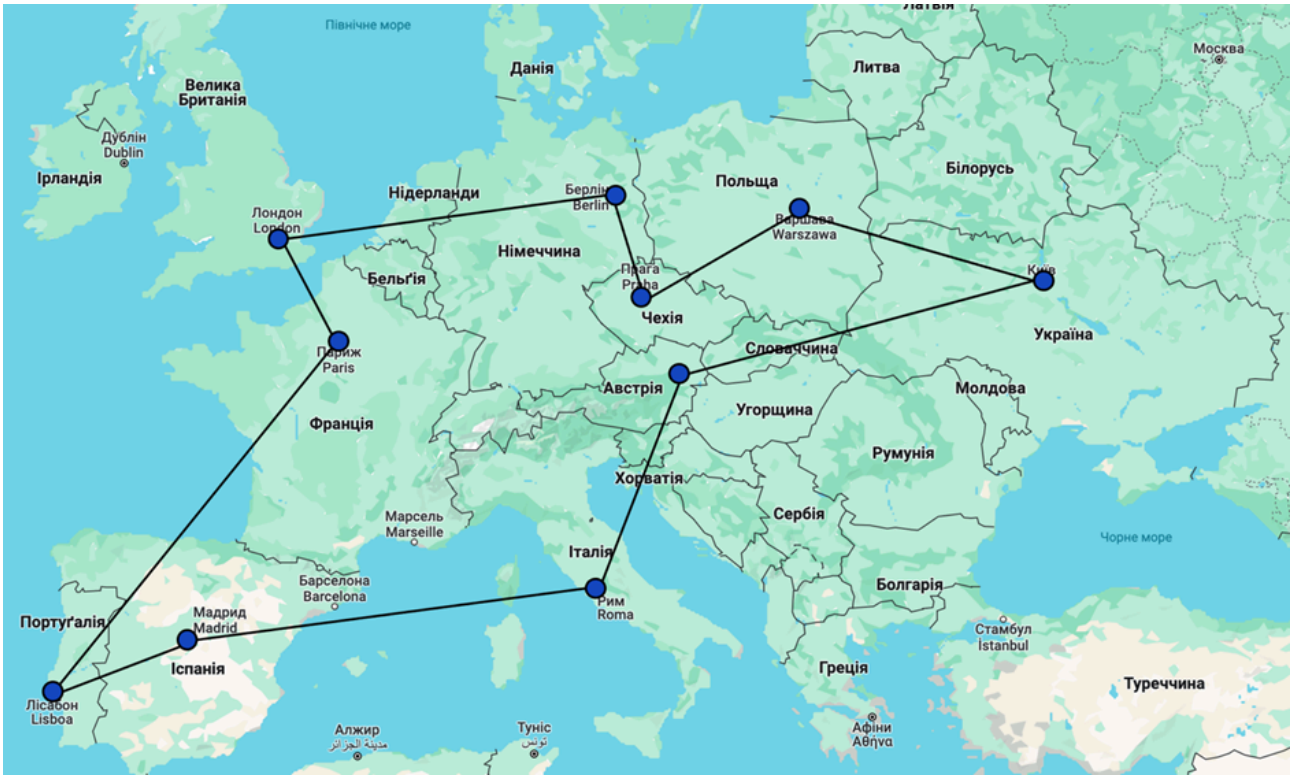

Рис. 1 – результат роботи геометричного алгоритму без урахування існуючих доріг

Як бачимо з рисунку 1, незважаючи на те, що геометричний алгоритм будує найкоротший шлях, він не є ідеальним і його неможливо використати в реальному житті, адже неможливо поїхати, наприклад з Мадриду до Риму напряму через те, що там є Середземне море, отже потрібно вдосконалити алгоритм так, щоб він міг урахувати під час обчислень найкоротших шляхів існуючі дороги між містами.

Для реалізації задуму використаємо бібліотеку MapKit [4] для мови програмування swift. Вона надасть нам дані про існуючі дороги, а геометричний

алгоритм разом із цими даними тепер зможе проаналізувати отримані параметри і видати правильне рішення для наземних видів транспорту. Тепер геометричний алгоритм діятиме за такою схемою:

1) Аналіз всіх координат всіх користувачів міст, через які він хоче побудувати маршрут.

2) Побудова маршруту лише за координатами міст.

3) Порівняння збудованого маршруту з існуючою реальною дорогою, яка з'єднує обрані два міста.

4) Перестановка та перебудова маршруту, якщо це необхідно.

Результат роботи геометричного алгоритму із застосуванням даних про існуючі дороги представлений на рисунку 2.

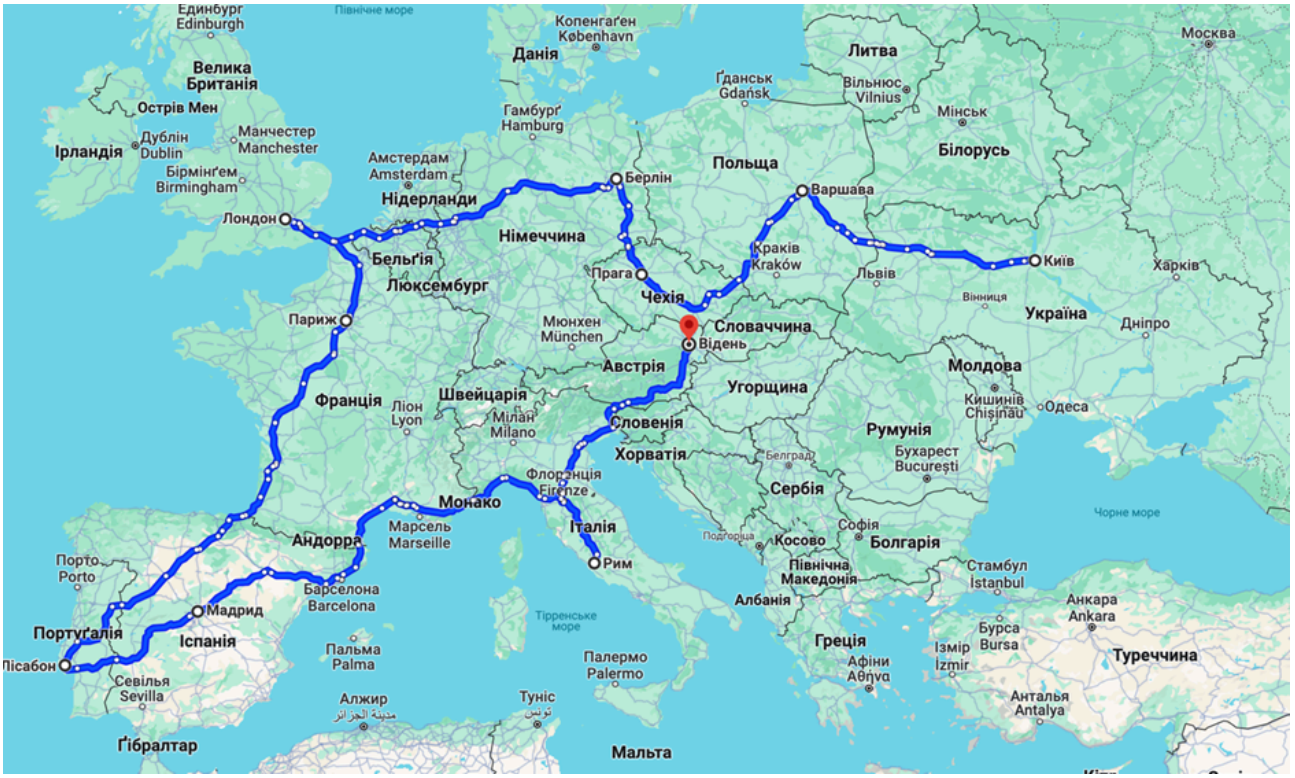

Рис. 2 – результат роботи геометричного алгоритму з урахуванням існуючих доріг

Як бачимо тепер геометричний алгоритм цілком правильно будує маршрути для наземних видів транспорту, і навіть з рисунку 2 видно, що не одне місто у результаті не з'єднує пряма лінія, отже застосування геометричного алгоритму без використання MapKit нема сенсу. Часові витрати на побудови маршруту значно не збільшилися(Рис. 3), алгоритмічна складність не змінилася, а точність побудови маршруту зросла і тепер підрахунок найкоротшого маршруту для існуючих міст можна виконати точніше і не повільніше, ніж до впровадження MapKit.

	Алгоритм повного перебору			Покращений геометричний алгоритм з використання MapKit		
	час виконання, с.	отриманий шлях	похибка	час виконання, с.	отриманий шлях	похибка
n = 10	0,36288	8893	0%	0,000037	8893	0%
Найкращий шлях = 8893						

Рис. 3 – результати порівняння алгоритму повного перебору, який використовує MapKit і покращеного геометричного алгоритму, який використовує MapKit

Висновки. У цій роботі запропоновано вдосконалення геометричного алгоритму для вирішення задач оптимізації транспортного руху, який використовує дані про дороги. Дане удосконалення дозволяє будувати маршрути для всіх видів наземного транспорту коректно, тому що тепер як відстань між двома містами використовується не пряма лінія, а складна фігура, що дозволяє вимірювати відстані ще з меншою похибкою. Також це удосконалення практично не змінило час виконання алгоритму, а значить алгоритм зможе швидко справлятися з великими обсягами даних ефективно, як раніше. Мобільний додаток, розроблений з використанням даного алгоритму, може бути успішно використаний у будь-яких логістичних проблемах, де швидкість побудови маршрутів та низька похибка при обчисленні шляху є важливими критеріями.

Список інформаційних джерел

1. Principles of Flight – The Four Forces Simply Explained [Електронний ресурс] – Режим доступу: https://selectaviation.com/principles-of-flight-the-four-forces-simply-explained/
2. Development of Ship Route-Planning Algorithm Based on Rapidly-Exploring Random Tree (RRT*) Using Designated Space [Електронний ресурс] – Режим доступу: https://www.mdpi.com/2077-1312/10/12/1800
3. Geometric design of roads [Електронний ресурс] – Режим доступу: https://en.wikipedia.org/wiki/Geometric_design_of_roads
4. Framework MapKit [Електронний ресурс] – Режим доступу: https://developer.apple.com/documentation/mapkit/