за 1 год. А оскільки сумарно за результатом розрахунків ми отримали 7 сонячних батарей, що потужніші в 6 та 9 разів, то провівши відповідні розрахунки, можемо вважати, що отримана кількість енергії є правдоподібною.

Список використаних джерел

1. IRENA: Vast majority of new renewables in 2022 have lower costs than fossil fuel–fired electricity. URL: https://balkangreenenergynews.com/irena-vast-majority-of-new-renewables-in-2022-have-lower-costs-than-fossil-fuel-fired-electricity/

2. Guide to Solar Optimization. URL: https://arkresources.com.au/guide-to-solar-optimization/

3. Simplex Method. URL: https://www.britannica.com/topic/simplex-method

УДК 004.021

Чемес В. С., здобувач 2 курсу спеціальності 122 Комп'ютерні науки,

Вєтров О. С., старший викладач кафедри прикладної математики та кібербезпеки

ПОРІВНЯННЯ АЛГОРИТМІВ ПОШУКУ НАЙКОРОТШОГО ШЛЯХУ

Донецький національний університет імені Василя Стуса, м. Вінниця

Однією з ключових задач, пов'язаних із пошуком найефективніших шляхів у графах та мережах, є пошук найкоротшого шляху між двома вершинами графу. Для вирішення цієї проблеми було розроблено різноманітні алгоритми, серед яких важливе місце займають алгоритми A* та Дейкстри.

Алгоритм Дейкстри – це метод для знаходження найкоротших шляхів у зважених графах із невід'ємними вагами ребер. Винахідником є голландський математик і інженер Едсгер Дейкстра (1959). Використовується для задач маршрутизації в телекомунікаційних мережах.

Основна ідея алгоритму полягає в систематичному відстеженні та оновленні найкоротших шляхів до кожної вершини з початкового вузла графу. Алгоритм використовує вагу шляху, що розповсюджується через ребра, обираючи вершину з найменшою вагою як поточний вузол. Оновлює ваги сусідніх вершин, враховуючи вагу ребра та активну вершину.

Кроки алгоритму:

1. Ініціалізація (встановлення ваги початкового вузла як 0, інші – як нескінченність).

2. Вибір вузла (обирається вершина з найменшою вагою як поточний вузол).

3. Оновлення ваги (обчислення ваг нових шляхів до сусідніх вершин. Оновлення ваги, якщо нова вага менша від поточної).

4. Позначення вузла (позначення поточного вузла як відвіданого).

5. Повторення (повторення кроків 2–4, доки не будуть відвідані всі вершини або досягнутий кінцевий вузол).

Алгоритм завершується, коли всі вершини відвідані або досягнутий кінцевий вузол. Важливо, що він працює лише для графів із невід'ємними вагами ребер і не обробляє від'ємні ваги [1].

Алгоритм A* є комбінацією точних методів та евристичних оцінок для ефективного пошуку найкоротшого шляху в графах. Його основна ідея полягає в тому, щоб враховувати як вартість досягнення поточної вершини, так і передбачувану вартість досягнення цільової вершини.

Кроки алгоритму A* включають:

1. Ініціалізація (встановлення початкової і цільової вершин, визначення початкового шляху).

2. Створення списків (відкритий список для неперевірених вершин та закритий список для вже оброблених).

3. Вартості (розрахунок вартостей для кожної вершини на основі вартості досягнення та евристичної оцінки до цільової вершини).

4. Вибір вершини (вибір вершини з найменшою загальною вартістю з відкритого списку).

5. Перевірка сусідів: (розгляд сусідніх вершин, оновлення вартостей, якщо новий шлях коротший).

6. Додавання та видалення вершин (додавання поточної вершини до закритого списку та видалення з відкритого).

7. Повторення (продовження кроків до досягнення цільової вершини або опорожнення відкритого списку).

8. Відновлення шляху (якщо цільова вершина досягнута, відновлення найкоротшого шляху від цільової до початкової вершини).

A* є ефективним та універсальним алгоритмом для пошуку шляхів у різних галузях застосування [2].

Тепер переглянемо програмну реалізацію кожного алгоритму на мові програмування Python. Ми розглянемо кожен з алгоритмів, який відображає найкоротший можливий шлях проходження від початкової точки (0, 0) до кінцевої точки (499, 499) і також відображає випадкові перешкоди, кількість яких становить 300.

Результати виконання кожного з алгоритмів можна розглянути на рис. 1 та 2 відповідно.

Execution Time: 6.765911102294922 seconds – це час виконання алгоритму Дейкстри.

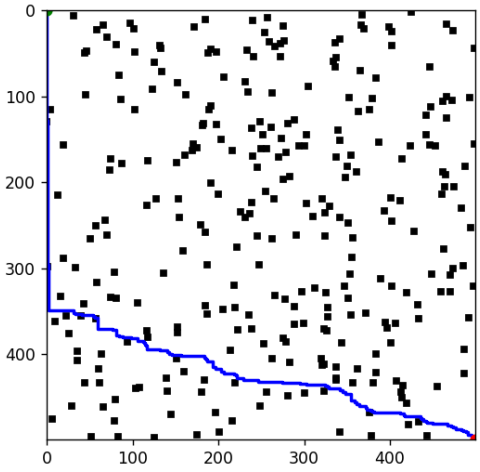

Рисунок 1. Шлях, утворений алгоритмом Дейкстри

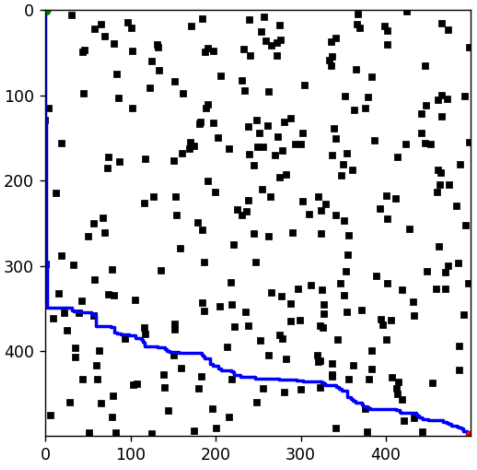

Рисунок 2. Шлях, утворений A-алгоритмом*

Execution Time: 0.8401174545288086 seconds – це час виконання A*-алгоритму. Як ми можемо помітити, A*-алгоритм виконується швидше за алгоритм Дейкстри у ≈8,05 разів. Алгоритм A* може бути більш швидким, порівняно з алгоритмом Дейкстри, у випадках, коли є евристична інформація про можливі вартості шляхів до кінцевої точки.

Основна відмінність між алгоритмами полягає в тому, що алгоритм A* додатково використовує евристичну функцію для оцінки залишкової вартості шляху до кінцевої точки. Ця евристична інформація дає змогу алгоритму A* більш ефективно обирати напрямок пошуку, спрямовуючись до очікувано більш «перспективних» шляхів.

Порівняно з алгоритмом Дейкстри, який розглядає всі можливі шляхи, алгоритм A* може прискорити пошук, концентруючись на областях, які, ймовірно, приведуть до оптимального шляху. Це особливо корисно у великих графах або задачах із великою кількістю можливих шляхів [3, 4].

Висновки. Алгоритм A* та алгоритм Дейкстри є ефективними методами пошуку найкоротших шляхів у графах, проте вони використовують різні стратегії для досягнення цієї мети.

Алгоритм Дейкстри гарантує знаходження найкоротшого шляху у графі з невід'ємними вагами ребер і розглядає всі можливі шляхи від початкової точки до кінцевої, вибираючи завжди найкоротший на поточний момент, але може бути витратним за часом, особливо у великих графах.

Алгоритм A* використовує евристичну інформацію для оцінки залишкової вартості шляху до кінцевої точки і спрямовує пошук у напрямку, який, імовірно, приведе до оптимального шляху, що може підвищувати ефективність у великих графах, але вимагає уважного вибору та налаштування евристичної функції для досягнення оптимальності.

Загалом вибір між алгоритмом A* та алгоритмом Дейкстри залежить від конкретних умов задачі. Якщо є можливість коректно визначити евристичну інформацію і вона допомагає зменшити простір пошуку, то A* може бути більш вигідним. У випадках, коли точна інформація важлива, алгоритм Дейкстри може бути більш оптимальним вибором.

Список використаних джерел

1. Dijkstra's algorithm. URL: https://en.wikipedia.org/wiki/Dijkstra%27s_algorithm (дата звернення: 27.11.2023).

2. A* search algorithm. URL: https://en.wikipedia.org/wiki/A*_search_algorithm (дата звернення: 28.11.2023).

3. Dijkstra's Algorithm. URL: https://www.programiz.com/dsa/dijkstra-algorithm (дата звернення: 30.11.2023).

4. A* Search Algorithm. URL: https://www.geeksforgeeks.org/a-search-algorithm/ (дата звернення: 01.12.2023).

УДК 311

Юстименко Є. А., здобувач 3 курсу спеціальності 122 Комп'ютерні науки,

Волонтир Л. О., канд. техн. наук, доцент, доцент кафедри інформаційних технологій

МОЖЛИВОСТІ ВИКОРИСТАННЯ РАНГОВОЇ КОРЕЛЯЦІЇ В РЕАЛЬНОМУ ЖИТТІ

Донецький національний університет імені Василя Стуса, м. Вінниця

Зв'язок між різними змінними є ключовим елементом аналізу даних у багатьох галузях. У світі різноманітних даних, коли розподіл даних може бути неоднорідним або відсутній нормальний характер, методи кореляції є важливим інструментом для виявлення та вимірювання ступеня зв'язку між змінними [1].

Один із таких методів – рангова кореляція – відіграє ключову роль у визначенні ступеня взаємозв'язку між змінними, які можуть мати порядковий характер або не відповідати умовам нормального розподілу. Використання рангових коефіцієнтів кореляції дає змогу робити висновки про взаємозв'язок між змінними навіть у випадках, коли інші методи кореляції втрачають свою ефективність.

Розглянемо основні можливості використання рангової кореляції в реальному житті [2]:

- У медичних дослідженнях рангова кореляція відіграє важливу роль у визначенні зв'язків між різними параметрами, як-от ефективність лікування,