DOI: 10.25140/2411-5363-2024-2(36)-151-161
УДК 004.62

Владислав Сергійович Зетченко[1], Артем Олександрович Задорожній[2]

[1]аспірант кафедри інформаційних технологій та програмної інженерії
Національний університет «Чернігівська політехніка» (Чернігів, Україна)
E-mail: zetchenkovlads@gmail.com. **ORCID:** https://orcid.org/0009-0002-9925-8628. **ResearcherID:** ITW-2651-2023

[2]кандидат технічних наук, доцент кафедри інформаційних технологій та програмної інженерії
Національний університет «Чернігівська політехніка» (Чернігів, Україна)
E-mail: zaotroy@gmail.com. **ORCID:** https://orcid.org/0000-0002-3424-7293. **ResearcherID:** F-6358-2016

ОПТИМІЗАЦІЯ МАРШРУТУ ГРОМАДСЬКОГО ТРАНСПОРТУ ЗА ДОПОМОГОЮ АЛГОРИТМУ A*

Стаття присвячена актуальній проблемі оптимізації маршрутів громадського транспорту у великих містах з використанням алгоритму A. Зазначено зростання популяції та вимог до транспортної інфраструктури, що підкреслює важливість забезпечення оптимальних маршрутів для зменшення транспортних заторів та викидів шкідливих речовин. У статті проаналізовано останні дослідження в цій сфері, які використовують різні алгоритми, такі як генетичні алгоритми, метаевристичні методи та штучні нейронні мережі. Представлена у статті інформація має оглядовий характер.*

Наведені статистичні дані дорожнього руху за період з 2023 по 2024 рр.

Ключові слова: *оптимізація маршрутів; громадський транспорт; алгоритм A*.*

Рис.: 3. Табл.: 2. Бібл.: 10.

Актуальність теми дослідження. Оптимізація маршрутів громадського транспорту є актуальною проблемою в сучасному світі, де великі міста стикаються зі збільшенням популяції та зростаючими вимогами до ефективності та доступності транспортної інфраструктури. Забезпечення оптимальних маршрутів може покращити рух транспорту, зменшити час подорожі та витрати палива, а також зменшити транспортні затори та викиди шкідливих речовин в атмосферу. У зв'язку з цим дослідження у сфері оптимізації маршрутів громадського транспорту є дуже актуальними.

Постановка проблеми. Головною проблемою, яка стоїть перед владою і транспортними компаніями, є забезпечення оптимальних маршрутів для громадського транспорту з урахуванням різноманітних факторів, таких як трафік, попит на перевезення та обмеження інфраструктури. Ця проблема стає ще складнішою у великих містах зі складною мережею транспортних маршрутів та великою кількістю пасажирів.

Аналіз останніх досліджень і публікацій. Останніми роками було проведено багато досліджень у сфері оптимізації маршрутів громадського транспорту за допомогою різних алгоритмів та підходів. Деякі дослідження фокусувалися на використанні генетичних алгоритмів, інші – на метаевристичних методах, таких як алгоритми мурашиної колонії або віджучення, а також на застосуванні штучних нейронних мереж. Ці дослідження надали цінний внесок у зрозуміння проблеми та розвиток ефективних методів оптимізації.

Виділення недосліджених частин загальної проблеми. Незважаючи на значний прогрес у цій області, існують деякі аспекти, які залишаються недослідженими або потребують подальшого дослідження. Наприклад, більш глибоке вивчення взаємодії між різними видами транспорту та врахування їх взаємодії при оптимізації маршрутів. Також важливо дослідити вплив різних стратегій планування маршрутів на загальну ефективність системи громадського транспорту.

Мета статті. Метою цієї статті є систематизація та огляд сучасного стану досліджень у галузі оптимізації маршрутів громадського транспорту з використанням алгоритму A*. Вона має на меті зрозуміти й підкреслити значення використання цього алгоритму в контексті вдосконалення систем громадського транспорту для більш ефективного та зручного пересування мешканців міст.

© В. С. Зетченко, А. О. Задорожній, 2024

Виклад основного матеріалу.

Порівняльний аналіз. Алгоритм A* є популярним методом для пошуку оптимальних маршрутів у графах, таких як дорожні мережі чи ігрові карти [1]. Він поєднує переваги жадібного пошуку та пошуку за першим найкращим шляхом, використовуючи евристичну функцію для оцінки вартості шляху. У цьому розділі порівнюються A* з кількома схожими алгоритмами оптимізації маршрутів.

Алгоритм Дейкстри. Алгоритм Дейкстри призначений для пошуку найкоротшого шляху від початкової вершини до всіх інших вершин у графі з не негативними вагами ребер.

Переваги:

- Гарантовано знаходить найкоротший шлях.
- Працює для всіх графів із не негативними вагами ребер.

Недоліки:

- Працює повільніше, ніж A*, якщо використовувати без евристики, оскільки розглядає всі можливі шляхи.
- Не використовує додаткову інформацію про цільову вершину.

Порівняння з A*: A* більш ефективний завдяки використанню евристичної функції, яка направляє пошук в бік цілі.

Алгоритм Дейкстри є окремим випадком A*, коли евристична функція завжди дорівнює нулю. Це означає, що A* при нульовій евристиці працює як Дейкстра, але з додатковими перевагами можливості адаптуватися до конкретних цілей завдяки евристичній оцінці.

Жадібний пошук за першим найкращим шляхом. Жадібний пошук за першим найкращим шляхом використовує лише евристичну функцію для вибору наступної вершини для дослідження, ігноруючи вартість шляху, пройденого до неї.

Переваги:

- Часто працює швидше, ніж A*, оскільки досліджує менше вершин.
- Простий у реалізації та зрозумілий.

Недоліки:

- Не гарантує знаходження оптимального шляху.
- Може застрягти в локальних мінімумах, якщо евристика не є оптимістичною.

Порівняння з A*: A* завжди гарантує знаходження найкоротшого шляху, якщо евристика допустима (не переоцінює реальну відстань до цілі).

Жадібний пошук за першим найкращим шляхом не враховує пройдену відстань, тоді як A* враховує як пройдену відстань, так і очікувану відстань до цілі, що робить його більш надійним для знаходження оптимальних маршрутів.

Пошук у ширину. Пошук у ширину є методом обходу графа, де кожен вузол досліджується рівномірно на всіх рівнях до знаходження цілі.

Переваги:

- Гарантує знаходження найкоротшого шляху в графах без ваги або з однаковою вагою ребер.
- Простий у реалізації.

Недоліки:

- Не підходить для графів з різною вагою ребер.
- Вимагає значних обчислювальних ресурсів та пам'яті для великих графів.

Порівняння з A*: A* ефективніший у графах з різною вагою ребер завдяки використанню евристичної функції.

Пошук у ширину може бути неефективним для великих графів, тоді як A* може значно скоротити кількість досліджуваних вершин завдяки евристиці.

Отже, на цьому етапі вже можна зробити висновок про те, що Алгоритм A* є потужним і універсальним інструментом для пошуку оптимальних маршрутів, який об'єднує переваги декількох інших алгоритмів завдяки використанню евристичної функції. Порівняння з алгоритмами Дейкстри, жадібного пошуку за першим найкращим шляхом та пошуку у ширину показує, що A* є ефективнішим і надійнішим у багатьох випадках, особливо коли використовується правильна евристика.

Опис алгоритму. Алгоритм A* використовують для знаходження найкоротшого шляху між двома точками в графі [2]. У контексті систем громадського транспорту, використання алгоритму A* може виглядати таким чином:

Представлення графу:

- Визначення вершин та ребер: Граф моделюється, при цьому кожна зупинка чи станція представляє вершину, а маршрути між ними - ребра.
- Вага ребер: Кожному ребру графа присвоюється вага, яка може відображати час подорожі, вартість квитка або інші критерії, що впливають на вибір маршруту.

Початок та кінець маршруту:

- Вибір початкової та кінцевої точок: Користувач визначає, з якої зупинки він починає подорож і куди він хоче прибути.

Запуск алгоритму A*:

- Ініціалізація алгоритму: Початкова вершина вважається поточною, а решта вершин відомі лише як потенційні.
- Розрахунок маршруту: Алгоритм A* визначає найкоротший шлях між початковою та кінцевою точками, оцінюючи вагу кожної вершини з урахуванням ваг ребер та евристичної функції.
- Оновлення поточної вершини: Алгоритм обирає наступну вершину для розгляду, оновлює оцінки ваг для сусідніх вершин та продовжує процес досягнення кінцевої точки або досягнення кінця графа.

Вибір оптимального маршруту:

- Відновлення маршруту: після досягнення кінцевої точки або закінчення роботи алгоритму, відновлюється найкоротший шлях, який стає оптимальним маршрутом для подорожі.
- Врахування обмежень: додаткові обмеження, такі як пересадки або певні маршрути, можуть бути враховані для вибору найбільш відповідного маршруту для користувача.

Підрахунок часу та вартості:

- Обчислення часу подорожі: оцінка часу, який займе подорож за обраним маршрутом.
- Обчислення вартості: оцінка вартості квитка або проїзду за обраним маршрутом.

Відображення маршруту:

- Представлення на мапі: оптимальний маршрут може бути відображений на мапі або у спеціалізованому додатку, щоб користувач міг легко зорієнтуватися під час подорожі.

Таким чином, A* використовується в реальних системах громадського транспорту для планування оптимальних маршрутів та навігації [3]. Наприклад, у системах онлайн-карт, мобільних додатках для навігації та системах керування маршрутами автобусів та метро [4]. Використання алгоритму дає змогу забезпечити швидке та ефективне планування маршрутів, зменшити час подорожі та покращити загальний досвід користувачів транспорту.

Блок-схема алгоритму A* представлена на рис. 1.

Загалом, він є одним із найефективніших методів знаходження найкоротшого шляху у графі з вагами. Він поєднує в собі елементи жадібного та інформованого пошуку, що робить його ефективним для застосування у задачах оптимізації маршрутів громадського транспорту.

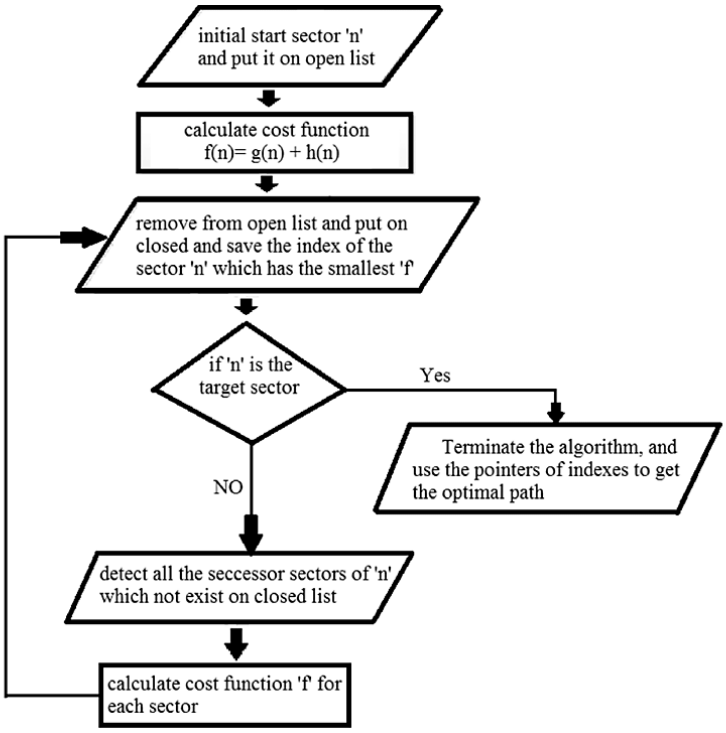

*Рис. 1. Блок-схема алгоритму A**

Джерело: розроблено авторами.

Ефективність. Алгоритм використовується в системах громадського транспорту з метою забезпечення оптимальних маршрутів для пасажирів [5]. Це важливо з огляду на ефективність та швидкість знаходження найкоротшого шляху в графі міста. Використання A* дозволяє зменшити час обчислень та забезпечити оптимальні результати, що робить його привабливим вибором для реалізації в системах масового транспорту. Крім того, цей алгоритм має властивість знаходити найкоротший шлях, що забезпечує ефективність та надійність маршрутів. Враховуючи складність міської інфраструктури та різноманітні умови пересування, ефективне планування маршрутів з використанням алгоритму A* є важливим елементом для покращення якості послуг громадського транспорту.

Оптимальність. A* відомий своєю оптимальністю у знаходженні найкоротших шляхів у графах з вагами.

Основним компонентом, який забезпечує оптимальність, є евристична функція або «попередній оцінювач», який дає оцінку вартості подорожі від поточної вершини до кінцевої. Ця функція дозволяє алгоритму пришвидшити пошук шляху, обмежуючи кількість розглянутих вершин. Зазвичай вона використовується для оцінювання вартості подорожі без врахування ваг ребер, тобто вона дає нижню межу вартості подорожі від поточної вершини до кінцевої.

Оптимальність алгоритму A* підтверджується математичною теорією та емпіричними дослідженнями. Його ефективність і надійність роблять його популярним інструментом для планування маршрутів у різних областях, включаючи громадський транспорт, логістику та штучний інтелект. Відмінна здатність алгоритму знаходити найкоротший шлях у графах з вагами робить його невід'ємним компонентом для вирішення багатьох проблем, де важлива оптимізація маршрутів і ресурсів.

Важливою властивістю оптимальності є гарантоване знаходження найкоротшого шляху при дотриманні відповідних умов. Це робить його ідеальним інструментом для оптимізації маршрутів громадського транспорту, де ефективність та економічність мають велике значення. Він дозволяє забезпечити мінімальні часи подорожі та мінімальні витрати ресурсів, що є критичними факторами для підвищення якості послуг транспортних систем.

Гнучкість. Гнучкість алгоритму A* полягає у його можливості адаптуватися до різних умов і вимог, що робить його ефективним інструментом для оптимізації маршрутів громадського транспорту. Основні аспекти гнучкості алгоритму A* включають:

Евристична функція: Алгоритм A* дозволяє використовувати різні евристичні функції для оцінки відстані між поточним вузлом і цільовим вузлом [6]. Це дозволяє адаптувати алгоритм до різних умов і вимог, таких як різні транспортні мережі або варіанти планування маршрутів.

Вага ребер: В алгоритмі A* можна використовувати різні ваги ребер для врахування різних факторів, таких як час подорожі, вартість квитків або комфортність маршруту. Це дозволяє враховувати різноманітні критерії оптимізації при плануванні маршрутів.

Адаптивність до змін: Алгоритм A* може ефективно працювати в умовах зміни параметрів графа, таких як зміна трафіку або розміщення пасажирів. Він може швидко адаптуватися до нових умов і знаходити оптимальні маршрути навіть у динамічному середовищі.

Гнучкість алгоритму A* робить його потужним інструментом для оптимізації маршрутів громадського транспорту, дозволяючи ефективно враховувати різноманітні умови та вимоги систем транспорту.

Масштабованість. Масштабованість алгоритму A* – це його здатність ефективно працювати зі збільшенням розміру вхідних даних, зокрема графів транспортних мереж. Однією з ключових переваг цього алгоритму є те, що він може застосовуватися до великих графів без значного збільшення часу обчислень.

Масштабованість досягається завдяки ефективному використанню пам'яті та оптимізації обходу графа. Алгоритм використовує структури даних, такі як пріоритетні черги та хеш-таблиці, для швидкого доступу до вершин графа та їх ваг. Це дозволяє зменшити час пошуку шляху і пам'яті, потрібної для зберігання інформації про граф.

Ще однією важливою характеристикою масштабованості алгоритму A* є його здатність працювати з графами різної складності. Він може ефективно враховувати різні фактори, такі як кількість вершин і ребер, ступінь зв'язності та розмір області пошуку. Це дозволяє використовувати його для планування маршрутів у великих містах зі складною транспортною інфраструктурою, де кількість вершин може бути дуже великою.

Це робить A* потужним інструментом для рішення різноманітних завдань планування маршрутів у великих транспортних системах. Алгоритм забезпечує ефективне врахування великої кількості даних і може працювати з графами будь-якої складності, що робить його важливим інструментом для розв'язання реальних проблем транспортного планування.

Опис даних. Для прикладу, в дослідженні використовується набір даних у форматі GeoJSON, який містить інформацію про зупинки громадського транспорту у місті Йорк [7]. Кожний об'єкт у наборі даних є географічним записом зупинки (на рис. 2 подана візуалізація даного набору даних).

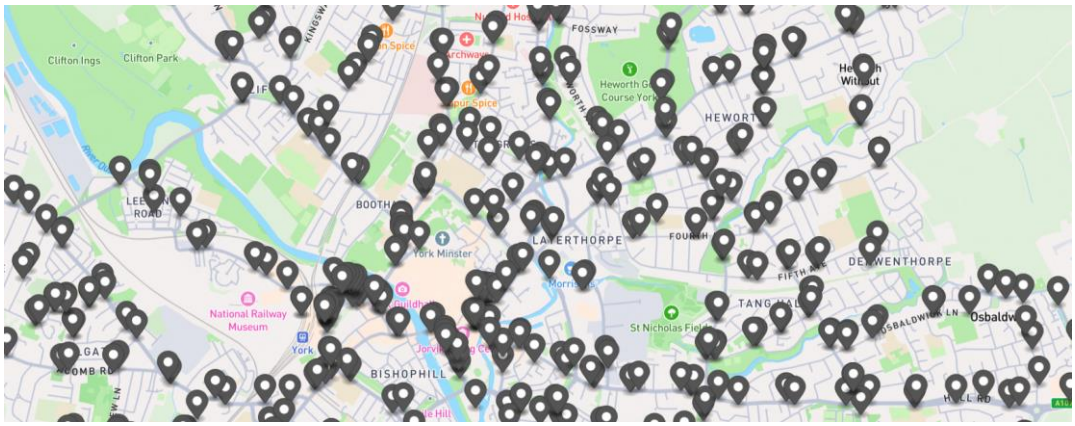

Рис. 2. Візуалізація вихідного набору даних

Джерело: розроблено авторами.

Задача полягає в тому, щоб знайти найкоротші маршрути для покриття всіх зупинок. Наступним кроком є побудова графу, в якому кожен вузол буде являти собою зупинку, а ребра будуть визначати можливі шляхи між зупинками [8].

Побудова графу. Для зчитування даних у форматі GeoJSON та обробки їх було використано засоби мови Python (повний лістинг програмного коду представлено на рис. 3).

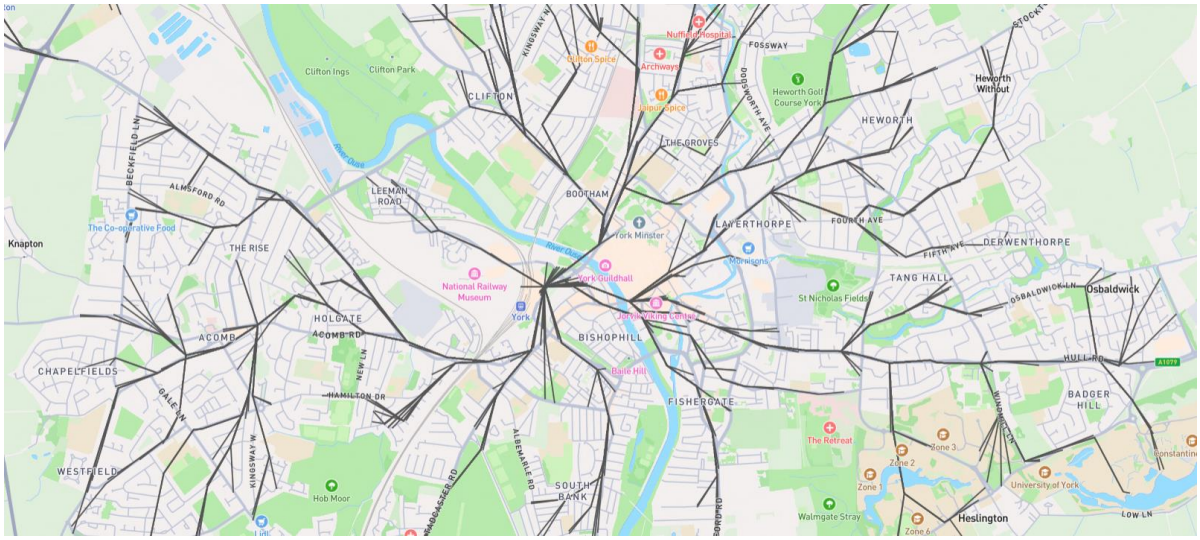

*Рис. 3. Візуалізація маршрутів A**

Джерело: розроблено авторами.

Лістинг програмного коду побудови графа:

```
# Зчитування даних з файлу GeoJSON
with open('stops.geojson', 'r') as file:
    data = json.load(file)

# Створення пустого графу
graph = {}

# Проходження через кожну зупинку
for stop in data['features']:
    stop_id = stop['properties']['stop_id'] # Ідентифікатор
    coordinates = tuple(stop['geometry']['coordinates']) # Координати
    graph[stop_id] = {'coordinates': coordinates, 'neighbors': {}} # Додавання зупинки до графу

# Пошук зв'язків між зупинками
for stop1_id, stop1_data in graph.items():
    for stop2_id, stop2_data in graph.items():
        if stop1_id != stop2_id:
            distance = geodesic(stop1_data['coordinates'], stop2_data['coordinates']).kilometers
            if distance < 0.5: # Приклад: встановлюємо максимальну відстань між зупинками
                graph[stop_id]['neighbors'][stop2_id] = distance # Додавання зв'язку до графу

# Збереження графу
with open('graph.json', 'w') as file:
    json.dump(graph, file, indent = 2)
```

Після завантаження та обробки даних з набору даних, перший крок у роботі з алгоритмом A* полягав у створенні графу. Кожна зупинка громадського транспорту була розглянута як окремий вузол у цьому графі. Цей процес вимагав перегляду кожного запису у наборі даних, отримання інформації про кожну зупинку та її координати, і подальше додавання цієї інформації у вигляді вузлів до графа [9].

Після створення вузлів було важливо встановити ребра, які представляли можливі зв'язки між зупинками громадського транспорту. Цей процес включав аналіз відстані між кожною парою зупинок та врахування максимальної відстані, яка вказувала на те, чи можна створити ребро між двома вузлами. Ця максимальна відстань відігравала ключову роль у формуванні структури графа, допомагаючи забезпечити оптимальне відображення зв'язків між зупинками та покриття всіх можливих маршрутів громадського транспорту.

Реалізація алгоритму A.* Для визначення оцінки відстані між двома зупинками була використана евристична функція: евклідова відстань між координатами цих зупинок. Це дозволило нам ефективно оцінювати найкоротший шлях між ними.

Лістинг реалізації алгоритму A* включав в себе використання евристичної функції для кожної пари зупинок, що дозволило шукати оптимальні маршрути у транспортній мережі з врахуванням їх географічного розташування.

Лістинг реалізації алгоритму A*:

```
# Реалізація алгоритму A*
def astar(graph, start, goal):
    # Ініціалізація відкритого та закритого списків
    open_set = []
    heappush(open_set, (0, start))
    came_from = {}
    g_score = {start: 0}
    f_score = {start: heuristic(graph, start, goal)}  # Оцінка шляху від початку до кінця через цю вершинудо кінця через цю вершину

    while open_set:
        _, current = heappop(open_set)
        if current == goal:
            return reconstruct_path(came_from, current)

        open_set.remove(current)
        if current in graph:
            for neighbor in graph[current]['neighbors']:
                tentative_g_score = g_score[current] + graph[current]['neighbors'][neighbor]
                if neighbor not in g_score or tentative_g_score < g_score[neighbor]:
                    checkCondition(neighbor, current) # Приклад: перевірка додаткової умови
                        came_from[neighbor] = current
                        g_score[neighbor] = tentative_g_score
                        f_score[neighbor] = g_score[neighbor] + heuristic(graph, neighbor, goal)
                        heappush(open_set, (f_score[neighbor], neighbor))

            return None  # Якщо неможливо знайти шлях

# Евристична функція для оцінки відстані між вершинами (в даному випадку, використовуємо евклідову відстань)
def heuristic(graph, node1, node2):
    if str(node1) in graph and str(node2) in graph:
        coord1 = graph[str(node1)]["coordinates"]
        coord2 = graph[str(node2)]["coordinates"]
        return math.sqrt((coord1[0] - coord2[0]) ** 2 + (coord1[1] - coord2[1]) ** 2)
    else:
        # Якщо одна з вершин відсутня у графі, повертаємо дуже велике значення, щоб не впливати на алгоритм
                                                    return float('inf')

# Приклад: функція перевірки умови
def checkCondition(stop_1, stop_2)
```

Для оптимізації алгоритму була застосована пріоритетна черга (heapq): використання пріоритетної черги для відкритого списку дозволяє ефективно керувати елементами з найменшою оцінкою (f_score), зменшуючи споживання пам'яті та прискорюючи доступ до необхідних елементів.

Необхідно звернути увагу на функцію checkCondition, так як вона виступає механізмом додаткової користувацької валідації, тобто може враховувати додаткові фактори, такі як напрямок руху (наприклад, в багатьох містах є вулиці з одностороннім рухом), який ніяк не зв'язаний із близькістю зупинок, а також обмеження щодо висоти або маси транспортного засобу (що є важливим, наприклад, для автобусів).

Отриманні результати були збережені у форматі GeoJSON і представлені на рис. 3.

Аналіз результатів. Дослідження було спрямоване на забезпечення покриття всіх зупинок громадського транспорту маршрутами з мінімальною кількістю повторень, використовуючи алгоритм A*. Загалом у графі містилося 1254 зупинки, які потребували ефективного планування для створення оптимальних маршрутів.

Результати роботи з пам'яттю системи:

- pmem = 59977728: Резидентний розмір набору, кількість пам'яті, яку зараз використовує процес в оперативній пам'яті (59,977,728 байт або приблизно 59,98 МБ).
- vms = 47349760: Віртуальний розмір пам'яті, загальний обсяг віртуального адресного простору, зарезервований для процесу (47,349,760 байт або приблизно 47,35 МБ).
- n_p_f = 39213: Кількість випадків помилок сторінок, які виникають, коли програма звертається до сторінки пам'яті, яка наразі не знаходиться в оперативній пам'яті.
- peak_paged = 208992: Піковий розмір кешу пам'яті, максимальна кількість кешованої пам'яті, яку використовував процес (208,992 байта).
- peak_nonpaged = 45600: Піковий розмір некешованого кешу пам'яті, максимальна кількість некешованої пам'яті, яку використовував процес (45,600 байт).

Нижче наведено дані (табл. 1), які характеризують результати роботи алгоритмів з ресурсами системи.

Таблиця 1 – Порівняльна характеристика ефективності аналогів A, %*

Алгоритм	pmem	vms	**n_p_f**	peak_paged	peak_nonpaged
Алгоритм Дейкстри	+0,33	-0,51	+2,07	+5,91	+37,10
Жадібний пошук	-0,89	-0,78	-1,05	+1,12	+3,17
Пошук в ширину	+1,89	+1,13	+19,41	+13,47	+78,63

Аналізуючи дані результати слід зазначити, що жадібний пошук хоч і впорався з задачею найшвидше, проте не зміг знайти оптимальний шлях (кількість отриманих маршрутів кратно перевищує результати інших алгоритмів. Також необхідно зазначити, що кількість маршрутів, яку знайшли інші представлені алгоритми, рівна). Беручи до уваги результати роботи алгоритмів, можна зробити висновок про те, що A* поступається лише в ефективності користування віртуального адресного простору.

Основною метою було забезпечення покриття всіх зупинок маршрутами. Використовуючи алгоритм A*, вдалося побудувати 891 маршрут, що охоплюють усі 1254 зупинки при використанні параметра максимальної відстані між зупинками зі значенням 500 метрів. З цих маршрутів 465 мають менше ніж 10 зупинок. Це свідчить про існування певних кластерів зупинок, ступінь віддаленості яких не дозволяє об'єднати їх у більші маршрути. Повне покриття всіх зупинок свідчить про ефективність алгоритму та правильність розрахунків, проведених у процесі планування.

Завдяки алгоритму A* вдалося досягти мінімальної кількості повторень маршрутів. Аналіз кількості маршрутів та зупинок у кожному маршруті показав, що алгоритм знайшов оптимальні шляхи для покриття всіх зупинок. Було виявлено, що 465 маршрутів ма-

ють менш як 10 зупинок, тоді як інші маршрути розподіляються на різні відстані та кількість зупинок. Це вказує на те, що алгоритм ефективно розподілив маршрути, враховуючи віддаленість та зв'язність між зупинками.

Слід зазначити, що кількість маршрутів оборотно-пропорційна до встановленого параметра – максимальна відстань між зупинками, який використовувався при побудові графа.

Таблиця 2 – Результати розрахунків

Максимальна відстань	Кількість маршрутів
0,5 км	891
1 км	386
2 км	174
5 км	71
10 км	49

Нині в місті Йорк функціонує 74 маршрути [10]. Проте ці маршрути оперуються більше ніж десятьма різними перевізниками й кожен діє у своєму кластері. Наприклад, перевізники 1–4 займаються виключно приміськими маршрутами. Деякі правила важко описати, проте алгоритм є потужним інструментом, який можна застосовувати точково, або ж модернізувати. Наприклад, у подальшому вдосконаленні системи можна додатково застосувати рекурсію: розділити будь-яке маршрутне покриття на кластери, оптимізувати маршрути в кожному кластері, а потім об’єднати ці кластери в маршрути, розглянувши їх як окремі зупинки.

Результати аналізу надали важливу інформацію для подальшого вдосконалення системи транспортних маршрутів. Було виявлено зупинки з високою кількістю повторень, що може бути корисним для оптимізації маршрутів та зменшення їхньої кількості. Ідентифікація таких зупинок дозволяє розробникам транспортних систем знаходити шляхи для подальшого поліпшення маршрутної мережі, забезпечуючи більш ефективне використання ресурсів та зменшення загального часу в дорозі для пасажирів.

Для виконання цього дослідження було використано кілька важливих методів та алгоритмів. Основним з них був алгоритм A*, який відомий своєю ефективністю у пошуку найкоротших шляхів. Евристична функція на основі евклідової відстані дозволила оцінювати відстані між зупинками з високою точністю. Також було застосовано методи аналізу графів для ідентифікації зупинок та їхніх сусідів, що дозволило ефективно будувати маршрути.

Висновки. Дослідження алгоритму A* у контексті оптимізації маршрутів громадського транспорту виявило його значну ефективність та потенціал для вдосконалення систем транспортних маршрутів. Застосування цього алгоритму дозволило досягти повного покриття всіх зупинок маршрутами з мінімальною кількістю повторень, що відображає його ефективність у знаходженні оптимальних маршрутів та забезпеченні ефективного та економічного транспортного обслуговування.

Важливою перевагою алгоритму A* є його оптимальність. Він гарантує знаходження найкоротшого шляху між двома точками у графі з вагами, що робить його ідеальним інструментом для планування маршрутів громадського транспорту. Оптимальність алгоритму забезпечує мінімізацію часу подорожі та оптимізацію витрат ресурсів, що є критичними факторами для підвищення якості обслуговування пасажирів.

Повне покриття усіх зупинок маршрутами є важливим кроком у вдосконаленні транспортних систем. Було побудовано 891 маршрут, охоплюючи усього 1254 зупинки, з яких 465 мають менше ніж 10 зупинок. Це свідчить не лише про ефективність алгоритму, але й про правильність розрахунків та можливість оптимізації маршрутів для покращення обслуговування пасажирів.

Крім цього, використання евристичної функції для оцінки відстані між зупинками дозволило ефективно оцінювати найкоротший шлях між ними. Це сприяло швидкому та ефективному плануванню маршрутів з урахуванням різних умов і обмежень, що забезпечує зручність та задоволення вимог пасажирів.

Отже, дослідження підтверджує успішне вирішення задачі оптимізації маршрутів громадського транспорту за допомогою алгоритму A*. Його ефективність та гнучкість роблять його важливим інструментом для подальшого вдосконалення систем транспортних маршрутів та забезпечення якісного обслуговування пасажирів.

Список використаних джерел

1. Джонсон, Д. Алгоритми оптимізації маршрутів / Д. Джонсон. – Кембридж: Массачусетський технологічний інститут, 2005.

2. Christian, B. Evolutionary Computation in Combinatorial Optimization / Christian Blum Gabriela Ochoa // Evolutionary Computation in Combinatorial Optimization : materials 14th European Conference, EvoCOP 2014 Granada, Spain, April 23-25, 2014. – Revised Selected Papers.

3. Bo, S. Optimization and Design Method of Feeder Bus System / S. Bo, W. Ming. – Wuhan: Scientific Research Publishing, 2020. – 200 с.

4. Моделювання і оптимізація маршрутів громадського транспорту з використанням алгоритмів штучного інтелекту // Журнал транспортної логістики. – Вроцлав, 2016.

5. Либерман, Л. Оптимізація маршрутів громадського транспорту з використанням алгоритмів інтелектуального аналізу даних / Л. Либерман. – Київ : Київський університет, 2019.

6. Гітіс, М. Моделювання та аналіз маршрутів громадського транспорту / М. Гітіс. – Лондон: Академічне видавництво, 2018.

7. Autobus zaustavljen u Yorku [Electronic resource] // European union. – Accessed mode: https://data.europa.eu/data/datasets/bus-stops3.

8. Ruisong, L. Data-Driven Bus Route Optimization Algorithm Under Sudden Interruption of Public Transport / L. Ruisong, W. Ning // Institute of Electrical and Electronics Engineers. – 2022. – № 10. – Pp. 5250-5263.

9. Analysis of modern ways of development of science and scientific discussions : The 10th International scientific and practical conference, Bilbao, ” (November 29 - December 02,2022). – Kansas : Primedia eLaunch LLC, 2022. – 606 p. – Accessed mode: https://isg-konf.com/wp-content/uploads/2022/11/Analysis-of-modern-ways-of-development-of-science-and-scientific-discussions.pdf.

10.Bus route and timetables – iTravel York [Electronic resource] // iTravel York. – Accessed mode: https://www.itravelyork.info/buses/bus-routes-and-journey-times/timetables.

References

1. Johnson, D. (2005). *Route optimization algorithms.* Cambridge: Massachusetts Institute of Technology.

2. Christian, B., Gabriela, O. (2014). *Evolutionary Computation in Combinatorial Optimization*. Granada: Revised Selected Papers.

3. Bo, S., Ming, W. (2020). *Optimization and Design Method of Feeder Bus System*. Wuhan: Scientific Research Publishing.

4. Modeling and optimization of public transport routes using artificial intelligence algorithms. (2016). *Journal of Transport Logistics.*

5. Liberman, L. (2019). *Optimization of public transport routes using algorithms of intelligent data analysis.* Kyiv: Kyiv University.

6. Hitis, M. (2018). *Modeling and analysis of public transport routes.* London: Academic Press.

7. European union. (2024). *Bus stops in the city of York.* https://data.europa.eu/data/datasets/bus-stops3.

8. Ruisong, L., Ning W. (2022). Data-Driven Bus Route Optimization Algorithm Under Sudden Interruption of Public Transport. *Institute of Electrical and Electronics Engineers*, *10*, 5250-5263.

9. International Science Group. (2022). *Analysis of modern ways of development of science and scientific discussions*. Kansas: Primedia eLaunch LLC.

10.Itravelyork. (2024). *Bus routes and timetables*. https://www.itravelyork.info/buses/bus-routes-and-journey-times/timetables.

Отримано 04.06.2024

TECHNICAL SCIENCES AND TECHNOLOGIES

UDC 004.62

Vladyslav Zetchenko[1], Artem Zadorozhniy[2]

[1]PhD Student of the Department of Information Technologies and Software Engineering
Chernihiv Polytechnic National University (Chernihiv, Ukraine)
E-mail: zetchenkovlads@gmail.com. **ORCID:** https://orcid.org/0009-0002-9925-8628. **ResearcherID:** ITW-2651-2023

[2] PhD in Technical Sciences, Associate Professor of the Department of Information Technologies and Software Engineering
Chernihiv Polytechnic National University (Chernihiv, Ukraine)
E-mail: zaotroy@gmail.com. **ORCID:** https://orcid.org/0000-0002-3424-7293. **ResearcherID:** F-6358-2016

PUBLIC TRANSPORT ROUTE OPTIMIZATION USING THE A* ALGORITHM

The article is focused on the actual problem of optimizing public transport routes in large cities using the A algorithm. In the context of a growing population and ever-increasing requirements for the efficiency and availability of transport infrastructure, ensuring optimal routes is crucial to reducing traffic congestion, shortening travel times and minimizing emissions.*

The paper reviews recent advances in the field, including various approaches to route optimization, such as genetic algorithms, metaheuristic methods, and applications of artificial neural networks. Special attention is paid to the informative review, which allows to systematize the current state of research and to determine the direction of further research in this important direction.

The purpose of the article is an in-depth analysis and systematization of the role of the A algorithm in improving public transport systems. Based on the collected information, its effectiveness compared to other methods and its potential for improving infrastructure in large cities are considered.*

Special attention is paid to the overview analysis, which allows to understand the current state and prospects for the development of this key field of scientific research.

The article not only presents a comprehensive overview of modern achievements, but also sets itself the task of promoting the A algorithm as a key tool for optimizing public transport routes, contributing to the increase in efficiency and convenience for residents of large cities.*

Traffic statistics for the period from 2023 to 2024 are provided.

***Key words**: route optimization; public transport; A* algorithm.*

Figures: 3. Tables: 2. References: 10.

Зетченко В. С., Задорожній А. О. Оптимізація маршруту громадського транспорту за допомогою алгоритму A*. *Технічні науки та технології.* 2024. № 2(36). С. 151-161.