УДК 621.311.1.018.3

О. Д. Долганенко; М. С. Широкопетлєва; О. О. Мазурова, канд. техн. наук, доц.; О. В. Вечур, канд. техн. наук, доц.; М. А. Долганенко

ДОСЛІДЖЕННЯ МЕТОДІВ ФОРМУВАННЯ ТУРИСТИЧНИХ ПОДОРОЖЕЙ

Метою статті є аналіз методів прийняття рішень в галузі туристичного планування подорожей. Мета туристичних подорожей - надати нам яскраві враження, і правильний вибір місця подорожі та добре спланований маршрут безсумнівно покращить враження від них. Метою роботи є підвищення ефективності підтримки користувачів при виборі напрямку для туристичної подорожі та плануванні послідовності міст для відвідування шляхом створення якісного програмного забезпечення в цій галузі. Для створення відповідного програмного забезпечення пропонується використовувати комбіновані методи прийняття рішень, які враховують індивідуальні особливості та переваги користувачів. Вивчалися методи, які підходять для вирішення розглянутого завдання, включаючи лінійну адитивну згортку з нормалізуючими факторами та ваговими коефіцієнтами, комбінаторику та задачу комівояжера. Було запропоновано вдосконалений алгоритм вибору та планування туристичної подорожі, який використовує комбінацію цих методів для отримання найкращого рішення для користувача. Було проведено експериментальне дослідження запропонованого алгоритму, яке дозволило перевірити його ефективність на всіх етапах. Експериментальне дослідження виконувалось на різних типах вхідних даних та враховувало різноманітні бажання та переваги користувачів. Для формування наборів даних для експериментів аналізувалися та порівнювалися різні джерела даних про міста, країни та туристичні атракції. Алгоритм був реалізований та інтегрований у мобільний додаток для Android з можливістю розширення, використовувалася методологія чистої архітектури. Запропоноване рішення може бути інтегроване в інші системи туристичного напрямку. За результатами можна зробити наступний висновок: запропонований удосконалений алгоритм та програмне рішення, розроблене на його основі, дозволяють враховувати переваги користувачів та покращувати досвід планування туристичних поїздок.

Ключові слова: *багатокритеріальна оптимізація, лінійна адитивна згортка, комбінації, задача комівояжера, планування поїздок.*

Вступ

Станом на сьогодні, через можливість легко користуватися розвинутими транспортними системами, туристи практично не мають технічних обмежень у подорожах. Кілька десятиліть тому це було не так легко через високі ціни на подорожі та небагато можливостей отримання віз, а також недоступності транспорту.

Головне питання мандрівників сьогодні не в тому, чи подорожувати, а куди саме подорожувати та яким чином. На це питання намагалися відповісти численні сервіси, такі як «Тріп Адвайзер», сайти туристичних агентств і навіть сервіси Google. Однак, як буде продемонстровано далі, усі ці системи не ідеально вирішують встановлену проблему.

Зазвичай турист вирішує куди поїхати, виходячи з попереднього особистого або рекомендованого досвіду (або його відсутності), уподобань, культурних інтересів, особистих рис, типів бажаних подорожей і, звичайно, ціни. Вищезазначені служби не враховують переваги багатьох користувачів і тому не пропонують найкращі варіанти. Крім того, є великий сегмент мандрівників, яким подобається постійно переміщатися цікавою країною/містом у пошуках нових вражень. Такі низькорівневі активності та побажання рідко покриваються наявними службами, і користувач сам має за них відповідати. Планування такої логістики поїздки загалом може бути дуже складним для більшості людей.

Мета статті полягає в тому, щоб зосередитися на аналізі та розробці автоматичних методів планування та оптимізації підготовки поїздки, а отже, покращенні досвіду для користувача.

Існує багато сайтів, присвячених подорожам. Коли користувачеві потрібно спланувати відпустку, перше, що запитує більшість сайтів, це місце призначення. Наприклад, сервіс «RomeToRio» – допомагає спланувати маршрути, в «Inspirock» – користувач може задавати пункти призначення, а сервіс пропонує повний список доступних активностей у містах, регіонах. А якщо користувач не має бажаних напрямків, є три варіанти.

Перший варіант – звернутися за допомогою до експерта-людини. Існують сайти планування, де потрібно ввести бажаний пункт призначення, ідеї, бюджет, потреби та інші переваги, а потім користувач з'єднується з експертом для планування подорожі. Таким самим чином працюють туристичні фірми.

Другий варіант – це використання пошуку рейсів із кількома фільтрами. Існують сайти з деякими загальними фільтрами, а також пошук. Отже, ці загальні фільтри надають загальні результати, які насправді не націлені на конкретного користувача. Або навпаки, деякі місця можуть бути пропущені через належність до іншої широкої категорії, але насправді бути хорошим варіантом за іншими критеріями.

Приклади включають "Great Escape", який є агрегатором розкладу рейсів і пошуковою системою з деякими фільтрами. Інший називається «Google Trips», де ключовими функціями є карти, можливість фільтрації за типом транспорту та загальними інтересами (популярні заходи, активний відпочинок, пляжі, музеї, історія, лижі тощо). А ще є сервіс під назвою «Tripmydream», де також можна застосувати деякі фільтри.

Третій варіант – використання послуг на основі випадкових пропозицій. Іноді це працює, але не для всіх і точно не кожного разу. Йдеться про сайти, які випадково пропонують подорожі та тури. Типовим прикладом є сайт під назвою "Eightydays.me", який є службою пошуку авіатурів для вибраного регіону.

У результаті, якщо користувач не знає куди поїхати, не хоче працювати з експертом, а широкі фільтри або випадкові пропозиції не допомагають, тоді потрібна вдосконалена система, яка допоможе цьому конкретному сегменту користувачів знайти та планувати подорож.

Огляд літератури та задача дослідження

На сьогодні існує достатньо багато систем, що підтримують користувачів під час обрання подорожі та проходження необхідних етапів, що супроводжують реалізацію цієї подорожі. У статті «Automated Travel Planning» [1] автора Macin наводиться алгоритм для планування маршруту подорожі, але недостатньо уваги приділено реалзіації алгоритму, особливо в аспекті користувацького інтерфейсу. Окрім цього досить обмежена інформація про ефективність системи в реальних умовах. В свою чергу, автори більш сучасної статті «Smart Travel Planning System Using Machine Learning – A Review» [2] розкривають досить актуальний підхід планування подорожей використовуючи машинне навчання, однак стаття має оглядовчий характер та розкриває штучний інтелект як інтерфейс до формування критеріїв та генерації опису подорожі, підкреслюється важливість якості вхідних даних та неможливість поетапної перевірки кроків роботи алгоритму через приховану сутність нейронних мереж.

З іншого боку, існують методи, які можна з точністю використовувати для пропонування подорожей на основі вподобань.

Наприклад, рішенням задачі багатокритеріальної оптимізації є згортки, які так само можна використовувати для фільтрації, рейтингу, пошуку найкращих варіантів на основі певних критеріїв.

Лінійна адитивна згортка обчислює скільки разів певна стратегія була оптимальною, вона не враховує кількісних показників значень критеріїв. Адитивна згортка особливо корисна, коли зниження оцінки за одним критерієм компенсується підвищенням оцінки за іншим критерієм або за кількома критеріями [3]. Легко побачити, що навіть якщо за деякими

критеріями оцінка дорівнює нулю, але інші показники критеріїв кращі, загальна оцінка все одно може бути задовільною. Лінійна адитивна згортка з нормалізуючими коефіцієнтами дозволяє працювати з кількісними критеріями, які мають різні одиниці, як у поточному випадку [4].

При зваженій лінійно-адитивній згортці кожен критерій множиться на власний ваговий коефіцієнт, а потім усі зважені критерії підсумовуються і утворюють зважену цільову функцію, значення якої інтерпретується як «коефіцієнт якості» отриманого рішення [5]. Отримана скаляризована функція максимізується в допустимому діапазоні обмежень.

Мультиплікативне згортання базується на постулаті: низький бал принаймні за одним критерієм призводить до загального низького значення. Отже, можна стверджувати, що мультиплікативна згортка базується на принципі справедливої компенсації відносних змін окремих критеріїв. У цьому випадку є два обмеження: сума вагових коефіцієнтів повинна дорівнювати 1, і кожен з вагових коефіцієнтів не повинен бути від'ємним. Вибір між адитивною та мультиплікативною згорткою визначається важливістю врахування абсолютних чи відносних змін значень окремих критеріїв.

Адитивна згортка обрана над мультиплікативною, тому що погані значення для одного з критеріїв не дискредитують весь варіант, як у випадку з мультиплікативною [6].

Мінімаксна згортка – найпростіший спосіб побудови узагальненого критерію (суперкритерію), заснований на застосуванні принципу мінімаксу. Кожен із критеріїв має свій вимір, і ці виміри зазвичай не збігаються. Тому спочатку необхідно стандартизувати всі наявні оцінки. Недоліком максимінної згортки є те, що вона враховує лише ті критерії, які дають найменші значення, усі інші критерії ігноруються. Через це максимінну згортку використовують не дуже часто, частіше використовують лінійну та мультиплікативну згортки. Але такий підхід завжди забезпечує результат, який гарантовано є мінімальним, нижче якого не будуть отримані інші значення [7].

Тоді, у разі планування поїздок, які складаються з кількох напрямків, необхідно буде комбінувати варіанти за допомогою комбінацій. Комбінація від n до k — це набір із k елементів, вибраних із набору розміром n, у якому порядок елементів не враховується. Тому комбінації, що відрізняються лише порядком елементів, але не складом, вважаються однаковими [8]. Якщо помножити комбінації, можна обчислити декартовий добуток [9].

Тоді, у разі планування поїздок, які складаються з кількох пунктів призначення, можливі маршрути потрібно отримати доступ для кожної комбінації:

Комівояжер – це одна з найпопулярніших задач комбінаторної оптимізації, яка полягає в пошуку найбільш вигідного маршруту, який хоча б один раз проходить через вибрані міста, а потім повертається в початкове місто.

При розв'язуванні задачі наводяться: критерій рентабельності маршруту (найкоротший, найдешевший, кумулятивний критерій тощо) та відповідні йому матриці відстаней, вартість. Як правило, зазначено, що маршрут повинен проходити через кожне місто лише один раз – у цьому випадку вибір здійснюється за допомогою гамільтонових шляхів. Гамільтонів шлях – це маршрут, який включає кожну вершину графа рівно один раз [10].

Задачу комівояжера можна представити у вигляді графа, тобто за допомогою вершин і ребер між ними. Вершини графа відповідають містам, а ребра між вершинами відповідають маршрутам сполучення між цими містами. Можна стверджувати, що вирішення проблеми полягає в знаходженні мінімально зваженого гамільтонового шляху в повному зваженому графі [11].

Одним із підходів до вирішення цієї проблеми є наївний підхід, який обчислює та порівнює всі можливі перестановки маршрутів для визначення найкоротшого рішення. Необхідно розрахувати відстань кожного маршруту, а потім вибрати найкоротший, який є найкращим рішенням.

Метод гілок і границь розбиває задачу, яку потрібно розв'язати, на кілька підзадач. Це

система для вирішення серії підзадач, кожна з яких може мати кілька можливих розв'язків і де рішення, вибране для однієї проблеми, може впливати на можливі розв'язки наступних підзадач. Для вирішення задачі необхідно вибрати початковий вузол, а потім задати обмежене дуже велике значення. Далі потрібно вибрати найдешевшу арку між невідвіданим і поточним вузлом, а потім додати відстань до поточної відстані. Процес слід повторювати, поки поточна відстань не стане меншою за обмеження. Якщо поточна відстань менша за обмежену відстань, обчислення можна завершити. Тепер можна додати відстань, щоб межа дорівнювала поточній відстані. Цей процес потрібно повторювати, поки не буде пройдено весь шлях [12].

Ключ до методу найближчого сусіда полягає в тому, щоб завжди відвідувати найближчий пункт призначення, а потім повертатися до першого міста, коли всі інші міста будуть відвідані. Щоб вирішити задачу продавця за допомогою цього методу, необхідно вибрати випадкове місто, знайти найближче невідвідане місто і поїхати туди. Після того, як усі міста відвідано, необхідно повернутися до першого міста [13].

Таким чином, було проаналізовано наявні методи та виявлено їх особливості.

На основі аналізу проблеми, аналогів та наявних методів було сформовано наступне завдання:

1. Використовувати методи комбінаторики для створення комбінацій та рішення задачі комівояжера для побудови маршрутів;
2. Створити алгоритм з використанням лінійної адитивної згортки з вагами та нормалізованими значеннями для оцінки варіантів;
3. Алгоритм має бути реалізований та інтегрований у програму Android із такими функціями:
   – встановлення персоналізованих налаштувань;
   – встановлення пріоритетів налаштувань;
   – відображення пропозицій.
4. Провести експеримент, дослідити ефективність використання обраних методів шляхом проведення експериментального дослідження алгоритму за допомогою розробленого додатку та формулювання рекомендації за результатами.

Запропонований алгоритм

На основі названих методів запропоновано новий алгоритм, який базується на кращих із зазначених методів.

Першим кроком є рейтинг і фільтрація країн і міст на основі вподобань користувачів за допомогою лінійної адитивної згортки з нормуючими множниками та ваговими коефіцієнтами. Список налаштувань може містити декілька параметрів. Чим більше користувач встановлює налаштувань, тим точнішою буде пропозиція.

Країни, а потім міста слід оцінювати таким чином: змінити значення, щоб відповідати одному принципу оптимізації, визначити множину Парето, нормалізувати дані, виконати згортку.

У результаті першого кроку отримуємо потенційний список країн та міст. Список відсортований за рейтингом розрахованим методом лінійної адитивної згортки відповідно до запитів користувача.

Другим кроком є об'єднання варіантів. Спочатку об'єднуємо країни в необхідну користувачу кількість країн для відвідування використовуючи комбінації з комбінаторики. А потім міста обраних країн будуть об'єднані таким же чином. Але тут буде використовуватися множення комбінацій.

Наприклад, одне місто з двох міст в одній країні та одне з шести міст в іншій країні призведе до дванадцяти комбінацій.

Результатом другого кроку є комбінації міст для відвідування в рамках однієї поїздки.

Третім кроком є оцінка можливих маршрутів для кожної комбінації у разі планування поїздок, які складаються з кількох пунктів призначення.

Необхідно встановити порядок відвідування міст для кожної комбінації з урахуванням як ціни, так і відстані. Ціна трансферу та відстань між містами будуть однаково важливими.

Результатом третього кроку є кращі варіанти напрямків для кожної комбінації з їх рейтингами.

Останнім кроком є оцінка всіх підготовлених комбінацій із застосуванням згортки з нормалізуючими коефіцієнтами.

Результатом останнього кроку є остаточна оцінка для кожного варіанту. Користувачу будуть представлені найкращі варіанти.

Особливим випадком алгоритму є ситуація коли потрібно обрати лише один пункт призначення. В такому разі буде застосовано лише перший крок із вибором країн і міст.

Діаграма активності [14] зображена на рис. 1 для представлення запропонованого алгоритму.

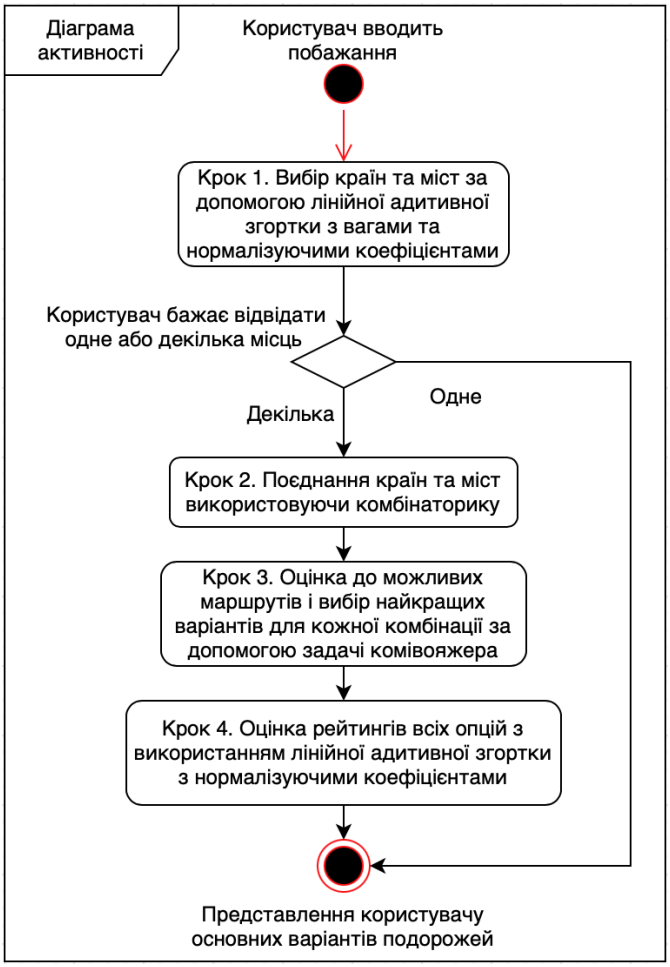

Рис. 1. Діаграма активності

Опис експерименту

Слідуючи згаданим вище крокам запропонованого алгоритму проводимо експеримент, щоб підтвердити концепцію. Однак задана ситуація і деякі кроки будуть спрощені для демонстраційних цілей.

Для першого кроку необхідно обрати можливі країни для відвідування користувача. Припустимо, що існує такий набір альтернатив: Франція, Італія, Чехія, Німеччина, Іспанія.

Значення для кожного критерію для кожної країни наведено в Таблиці 1 на основі значень, взятих з Інтернету.

Таблиця 1

Значення критеріїв для кожної країни

Країна	Температура	Кількість опадів	Наявність моря	Наявність гір	Ціна за день
Франція	26	60	1	1	65
Італія	31	24	1	1	80
Чехія	26	85	0	0	22
Німеччина	24	81	1	0	32
Іспанія	29	27	1	1	50

Потім необхідно привести всі дані до одного принципу оптимізації – або максимізації, або мінімізації. Температура, вартість і кількість опадів повинні бути мінімізовані. Отже, значення параметрів моря та гір потрібно змінити, щоб отримати той самий принцип оптимізації. Результати заміни значень на різницю між максимальним значенням і елементом відображено в Таблиці 2.

Таблиця 2

Приведені значення до одного принципу оптимізації

Країна	Температура	Кількість опадів	Наявність моря	Наявність гір	Ціна за день
Франція	26	60	0	0	65
Італія	31	24	0	0	80
Чехія	26	85	1	1	22
Німеччина	24	81	0	1	32
Іспанія	29	27	0	0	50

Перевірка множини Парето показує, що всі альтернативи не гірші за інші принаймні за одним критерієм, тобто жодна альтернатива не може бути вилучена.

Тепер необхідно нормалізувати значення. Використаємо наступну формулу методу нормалізації максимін:

$$x_{normalized} = \frac{x - \min(x)}{\max(x) - \min(x)}, \quad (1)$$

де *x* – значення що зараз обробляється, *min(x)* – мінімальне значення серед усіх, *max(x)* – максимальне значення серед усіх.

Результати розрахунків зображено в Таблиці 3.

Таблиця 3

Нормалізовані значення

Країна	Температура	Кількість опадів	Наявність моря	Наявність гір	Ціна за день
Франція	0.29	0.59	0	0	0.74
Італія	1	0	0	0	1
Чехія	0.29	1	1	1	0
Німеччина	0	0.93	0	1	0.17
Іспанія	0.71	0.05	0	0	0.48

Припустимо, користувач визначив наступні пріоритети для критеріїв:

- f1 – середня температура в липні – 0.2;
- f2 – кількість опадів в липні – 0.1;
- f3 – наявність моря – 0.3;
- f4 – наявність гір – 0.2;
- f5 – ціна за один день – 0.2.

Результати виконаних розрахунків на основі пріоритетів встановлених користувачем вище, представлені в Таблиці 4.

Таблиця 4

Розрахунки для кожної країни

Країна	Розрахунки
Франція	0.2*0.29 + 0.1*0.59 + 0.3*0 + 0.2*0 + 0.2*0.74 = 0.265
Італія	0.2*1 + 0.1*0 + 0.3*0 + 0.2*0 + 0.2*1 = 0.4
Чехія	0.2*0.29 + 0.1*1 + 0.3*1 + 0.2*1 + 0.2*0 = 0.658
Німеччина	0.2*0 + 0.1*0.93 + 0.3*0 + 0.2*1 + 0.2*0.17=0.327
Іспанія	0.2*0.71 + 0.1*0.05 + 0.3*0 + 0.2*0 + 0.2*0.48=0.243

Як наслідок, Іспанія та Франція мають кращий рейтинг, тому обираємо ці країни для демонстраційного експерименту.

Дотримуючись тих самих вищезазначених кроків, обираємо міста в кожній країні. Рейтинг іспанських міст такий: перше місце – Барселона, друге місце – Валенсія, третє місце – Гранада. Рейтинг французьких міст такий: перше місце – Марсель, друге – Монпельє, третє – Ніцца.

Переходимо до другого кроку запропонованого алгоритму, який об'єднує країни в необхідну користувачеві кількість країн для відвідування.

Оскільки в поточному випадку для демонстрації було обрано лише дві країни, доступна лише одна комбінація – Іспанія та Франція. Але потрібно згенерувати комбінації для вибраних міст.

У поточному експерименті користувач бажає відвідати одне місто з кожної з двох країн. Наразі з кожної країни обирається три міста, а потім буде створено дев'ять комбінацій за допомогою множення комбінацій.

Тоді результат зі списком комбінацій міст такий: Барселона в комбінаціях з кожним французьким містом, те саме для Валенсії та Гранади.

Третій крок – вирішення проблеми комівояжера. Спочатку необхідно надати значення цін і відстаней в обох напрямках.

Відстані між містами базуються на середніх значеннях з Інтернету. Значення міст, які не можуть бути в одному маршруті в поточному випадку, наприклад, Валенсія та Барселона опущені. Значення відстані між містами нормалізовано, щоб відстані можна було правильно співвіднести з цінами. Ціни на проїзд з одного міста в інше залежать від напрямку. Як і для відстаней, ціни також потрібно нормалізувати, щоб їх можна було правильно співвіднести з відстанями.

Результати додавання нормалізованих значень ціни та відстані для кожного міста відображені в Таблиці 5.

Таблиця 5

Сума нормованих значень ціни та відстані

Міста	Харків	Барселона	Валенсія	Гранада	Марсель	Монпельє	Ніцца
Харків	–	1.25	1.44	1.84	1.23	1.71	1.1
Барселона	1.22	–	–	–	0.2	0.45	0.15
Валенсія	1.4	–	–	–	0.15	0.55	0.58
Гранада	1.76	–	–	–	0.61	0.95	0.58
Марсель	1.16	0.14	0.12	0.54	–	–	–
Монпельє	1.51	0.4	0.48	0.9	–	–	–
Ніцца	1.08	0.12	0.5	0.62	–	–	–

На основі значень приведених вище вирішуємо задачу комівояжера та отримуємо наступний список найкращих маршрутів для кожної комбінації:

- П1: Харків, Марсель, Барселона, Харків – 2.52;
- П2: Харків, Барселона, Монпельє, Харків – 3.21;
- П3: Харків, Ніцца, Барселона, Харків – 2.44;
- П4: Харків, Марсель, Валенсія, Харків – 2.75;
- П5: Харків, Валенсія, Монпельє, Харків – 3.5;
- П6: Харків, Ніцца, Валенсія, Харків – 3;
- П7: Харків, Марсель, Гранада, Харків – 3.53;
- П8: Харків, Гранада, Монпельє, Харків – 4.3;
- П9: Харків, Ніцца, Гранада, Харків – 3.48.

Четвертим і останнім кроком алгоритму є оцінка всіх підготовлених комбінацій з використанням лінійної адитивної згортки з нормуючими множниками. Вага не розглядається, оскільки всі три критерії – рейтинг задачі комівояжера, рейтинг іспанського міста, рейтинг французького міста – однаково важливі.

Збираємо всі оцінки найкращих варіантів із попереднього кроку та помістимо їх у стовпець оцінки проблем комівояжера. Рейтинги міст беруться з першого кроку алгоритму. Усі перераховані рейтинги відображені в Таблиці 6.

У цьому випадку всі дані зводяться до мінімуму, тому на даний момент коригувати дані не потрібно. Потім потрібно нормалізувати значення за допомогою Формули 1. Результати відображаються в Таблиці 7.

Як уже було вказано вище, усі критерії однаково важливі, тому вагові коефіцієнти не застосовуються. Результати лінійної згортки: П1 – 0.04, П2 – 0.91, П3 – 1, П4 – 0.56, П5 – 1.57, П6 – 1.8, П7 – 1.59, П8 – 2.5, П9 – 2.56.

Таблиця 6

Три рейтинги для кожної комбінації

Варіанти	Оцінка задачі комівояжера	Рейтинг міста з Іспанії	Рейтинг міста з Франції
П1	2.52	1	1
П2	3.21	1	2
П3	2.44	1	3
П4	2.75	2	1
П5	3.5	2	2
П6	3	2	3
П7	3.53	3	1
П8	4.3	3	2
П9	3.48	3	3

Таблиця 7

Нормалізовані значення

Варіанти	Оцінка задачі комівояжера	Рейтинг міста з Іспанії	Рейтинг міста з Франції
П1	0.04	0	0
П2	0.41	0	0.5
П3	0	0	1
П4	0.16	0.5	0
П5	0.57	0.5	0.5
П6	0.3	0.5	1
П7	0.59	1	0
П8	1	1	0.5
П9	0.56	1	1

Таким чином найкращими варіантами подорожей є:

- П1: Харків, Марсель, Барселона;
- П4: Харків, Марсель, Валенсія;
- П2: Харків, Барселона, Монпельє.

Отже, було проведено експеримент виконуючи кроки запропонованого алгоритму.

Для оцінки ефективності та продуктивності запропонованого алгоритму було проведено кілька експериментів з використанням різних метрик. Експерименти проводилися на операційній системі Android, на пристрої Samsung Galaxy S8, який має процесор Exynos 8895 з 8 ядрами, 4 ГБ оперативної пам'яті та операційну систему Android 7.0 (Nougat).

Проведемо N = 5 запусків алгоритму на наборах даних різного обсягу (10, 20, 50 та 100 міст з однієї та різних країн) для отримання показників часу виконання що наведені в Таблиці 8.

Таблиця 8

Час виконання алгоритму

Кількість міст	Середній час виконання (с)	Мінімальний час (с)	Максимальний час (с)
10	0.5	0.45	0.55
20	1.2	1.1	1.3
50	3.5	3.3	3.7
100	7.8	7.5	8.1

Результати показали, що час виконання алгоритму знаходиться в межах, які задовольняють вимоги потенційних користувачів, навіть при збільшенні обсягу даних.

Моніторинг навантаження на центральний процесор під час виконання алгоритму показав, що використання ресурсів залишалося в допустимих межах. Під час експерименту не було запущено інших додатків, а система знаходилася в режимі "в літаку" для зменшення факторів впливу. Результати експерименту наведені в Таблиці 9.

Таблиця 9

Навантаження на ЦП

Кількість міст	Середнє навантаження на ЦП (%)	Мінімальне навантаження (%)	Максимальне навантаження (%)
10	14	11	17
20	15.5	13	18
50	17.5	15	20
100	27.5	26	29

Було виявлено, що обробка більших обсягів даних потребує більшого часу, але не призводить до критичного навантаження на систему.

Програмна реалізація

Алгоритм реалізовано за допомогою мови програмування Kotlin. Також була розроблена мобільна програма для Android як інтерфейс користувача, використовуючи Kotlin.

І алгоритм, і додаток були розроблені відповідно до найкращих принципів розробки програмного забезпечення, включаючи принципи чистої архітектури [15].

Для демонстрації користувацького досвіду у розробленому додатку додаємо скріншоти. Перший – початковий екран, на якому користувачі повинні вказати свої переваги щодо: країни, міста, клімату, опадів, гори поблизу, моря поблизу, максимальної вартості за день, початкового міста та виходячи з чого обрати маршрут – вартості , відстані або обидва.

Другий екран надає можливість встановити пріоритет уподобань, перетягнувши критерії, щоб встановити порядок від найбільш бажаних до найменш.

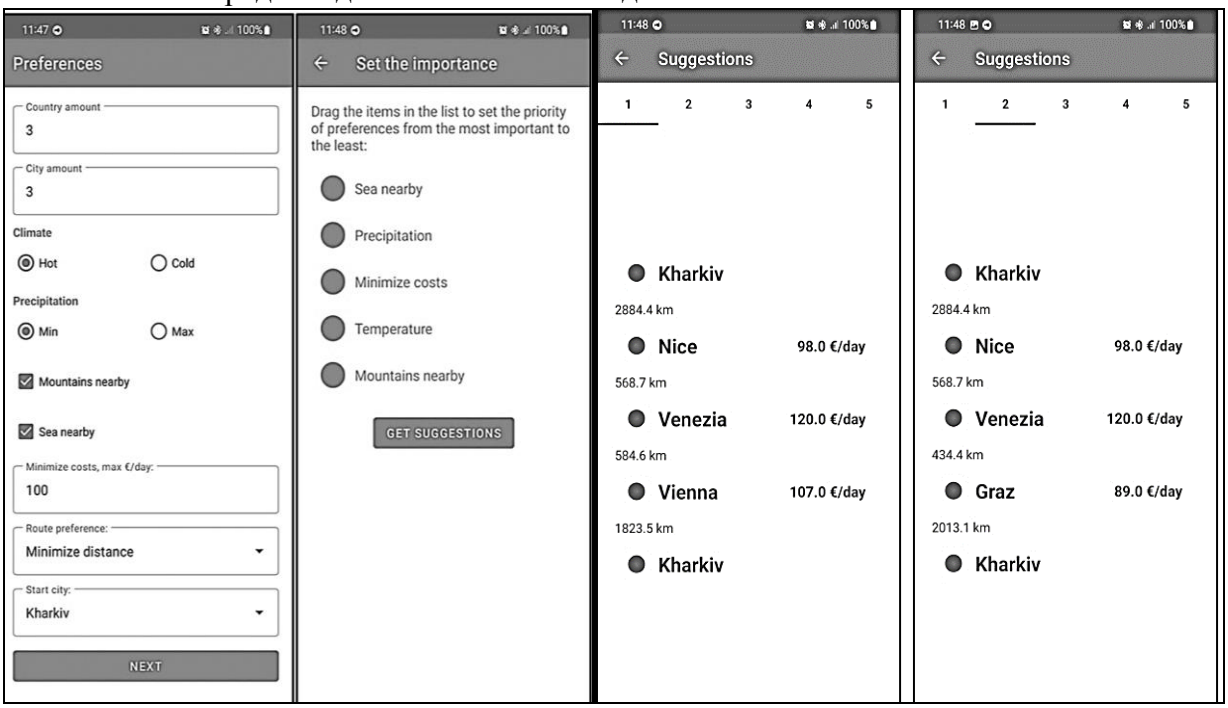

Рис. 2. Скріншоти екрана налаштувань користувача, налаштування пріоритету та запропоновані подорожі

Отже, було продемонстровано роботу програмної реалізації алгоритму пропонування подорожей.

Висновки

Проблема полягає в тому, що подорожі, особливо ті, які містять кілька міст, країн, які потрібно відвідати, потрібно пропонувати користувачам на основі їхніх уподобань. Чим більше налаштувань встановить користувач, тим точніше рішення буде запропоноване.

Було проаналізовано наявні сервіси, пов'язані з подорожами, і виявлено, що більшість із них не мають можливості пропонувати такі поїздки або сервіси мають інші проблеми. Переглянуто наявні методи та розроблено вдосконалений алгоритм, який базується на таких методах: лінійна адитивна згортка з нормалізуючими коефіцієнтами та вагами, методи комбінаторики, задача комівояжера. Після створення алгоритму було проведено експеримент, щоб довести, що запропонований алгоритм працює належним чином. Було проаналізовано швидкість роботи та навантаження на ЦП залежно від кількості вхідних даних та отримано задовільні результати, що сприяють гарному досвіду користувача.

У планах щодо подальших досліджень і вдосконалень є врахування всіх доступних способів транспортування для даного маршруту та пропонування точної кількості днів для кожного міста або іншого пункту призначення. Запропонований алгоритм можна використовувати на практиці, інтегрувавши його в програмне забезпечення туристичних фірм.

СПИСОК ЛІТЕРАТУРИ

1. Nagrodkiewicz Piotr. Automated Travel Planning / Piotr Nagrodkiewicz, Marcin Paprzycki // Research Gate. – 2005. – P. 1 – 12. DOI: 10.1109/ITICT.2005.12345.

2. Varghese A. Smart Travel Planning System Using Machine Learning – A Review / Alex Varghese, Amal Santhosh, Badhusha TE, Fayiz Jalal, Shameena EM // International Journal of Research Publication and Reviews. – 2023. – Vol. 4, №. 5. – P. 123 – 134. DOI: 10.12345/ijrpr.2023.123456.

3. Zgurovsky M. Z. Introduction. In: Combinatorial Optimization Problems in Planning and Decision Making. Studies in Systems, Decision and Control [Text] / Mykhailo Zgurovsky, Alexander Pavlov // Springer, Cham. – 2019. –

vol. 173. – P. 1 – 14. DOI: 10.1007/978-3-319-98977-8_1.

4. A study of optimization models for creation of artificial intelligence for the computer game in the tower defense genre / O. Mazurova, O. Samantsov, O. Topchii [et al.] // 2020 IEEE international conference on problems of infocommunications. science and technology (PIC S&T), Kharkiv, Ukraine, 6 – 9 October 2020. – Kharkiv, 2020. – P. 491 – 496. DOI: 10.1109/picst51311.2020.9468057.

5. Vakhania N. Multicriteria Optimization – Pareto-Optimality and Threshold-Optimality / N. Vakhania, F. Werner // IntechOpen. – 2020. – P. 106. DOI: 10.5772/intechopen.78897.

6. Misyurin S. Multicriteria Optimization of a Dynamic System by Methods of the Theories of Similarity and Criteria Importance / Sergey Misyurin // Mathematics. – 2021. – Vol. 9, №. 22. – P. 2854. DOI: 10.3390/math9222854.

7. Nelyubin A. P. Multicriteria choice based on criteria importance methods with uncertain preference information / A. P. Nelyubin, V. V. Podinovski // Computational Mathematics and Mathematical Physics. – 2017. – Vol. 57, № 9. – P. 1475 – 1483. DOI: 10.1134/s0965542517090093.

8. Sagan B. Combinatorics: The Art of Counting / B. Sagan. – Providence, United States, 2020. – 304 p.

9. Shao Z. Matching Book Embedding of the Cartesian Product of a Complete Graph and a Cycle / Shao Zeling, Liu Yanqing, Li Zhiguo // arXiv: Combinatorics. – 2020. – P. 89 – 97. DOI: 10.48550/arXiv.2002.00309.

10. Davendra D. Introductory Chapter: Traveling Salesman Problem – An Overview / Donald Davendra, Magdalena Bialic-Davendra // Novel Trends in the Traveling Salesman Problem. – 2020. – P.1 – 6. DOI: 10.5772/intechopen.94435.

11. Hossain S. An Efficient Solution to Travelling Salesman Problem using Genetic Algorithm with Modified Crossover Operator / Md Sabir Hossain // Emitter International Journal of Engineering Technology. – 2019. – Vol. 7, № 2. – P. 480 – 493. DOI: 10.24003/emitter.v7i2.380.

12. Mahfoudh S. S. A branch and bound algorithm for the porbabilistic traveling salesman problem / Soumaya Sassi Mahfoudh, Walid Khaznaji, Monia Bellalouna // 2015 IEEE/ACIS 16th International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD), Takamatsu. – Takamatsu, 2015. – P. 6567 – 6581. DOI: 10.1109/snpd.2015.7176284.

13. Towards Comparative Analysis Of Branch - And - Bound And Nearest Neighbour Algorithms / J. A. Ayoola, E. O. Asani, A. E. Okeyinka [et al.] // 2020 International Conference in Mathematics, Computer Engineering and Computer Science, Ayobo, Ipaja, Lagos, Nigeria, 18–21 March 2020. – Ayobo, 2020. – P. 1 – 5. DOI: 10.1109/icmcecs47690.2020.240901.

14. Osis J. Topological UML Modeling: An Improved Approach for Domain Modeling and Software Development / Janis Osis, Uldis Donins // Elsevier Science & Technology Books, 2017. – 276 p.

15. Martin R. Clean Architecture: A Craftsman's Guide to Software Structure and Design / Robert Martin – Pearson, 2017. – 432 p.

Стаття надійшла до редакції 14.05.2024.
Стаття пройшла рецензування 25.05.2024.

Долганенко Олександр Денисович **–** співробітник.
Lemberg Solutions.

Широкопетлєва Марія Сергіївна – старший викладач кафедри програмної інженерії, e-mail: marija.shirokopetleva@nure.ua.

Мазурова Оксана Олексіївна – доцент кафедри програмної інженерії, кандидат технічних наук, доцент.

Вечур Олександр Володимирович – доцент кафедри програмної інженерії, кандидат технічних наук, доцент.
Харківський національний університет радіоелектроніки.

Долганенко Марія Анатоліївна – співробітник.
Epam Systems.