ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Методика оптимізації комплектації багажу на основі динамічних показників»

на здобуття освітнього ступеня магістра
зі спеціальності 121 Інженерія програмного забезпечення
освітньо-професійної програми «Інженерія програмного забезпечення»

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ Михайло СКВОРЦОВ
(підпис)

Виконав: здобувач вищої освіти групи ПДМ-61
Михайло СКВОРЦОВ
Керівник: Людмила СОЛЯНИК
канд.наук
Рецензент:
науковий ступінь, вчене звання *Ім'я, ПРІЗВИЩЕ*

Київ 2026

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра Інженерії програмного забезпечення
Ступінь вищої освіти Магістр
Спеціальність 121 Інженерія програмного забезпечення
Освітньо-професійна програма «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри
Інженерії програмного забезпечення
_____ Ірина ЗАМРІЙ
«_____» _____ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Скворцову Михайлу Олександровичу

1. Тема кваліфікаційної роботи: «Методика оптимізації комплектації багажу на основі динамічних показників»

керівник кваліфікаційної роботи Людмила СОЛЯНИК, канд. наук

затверджені наказом Державного університету інформаційно-комунікаційних технологій від «30» жовтня 2025 р. №467.

2. Строк подання кваліфікаційної роботи «19» грудня 2025 р.

3. Вихідні дані до кваліфікаційної роботи: науково-технічна література, відкриті джерела динамічних показників, математична модель динамічної корисності, вимоги до оперативності рішення та обмеження ваги.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1. Дослідження та систематизація впливу зовнішніх динамічних показників на функцію цінності предметів багажу.
2. Розробка та математична формалізація моделі динамічної корисності та гібридного алгоритму комплектації багажу на основі адаптивної пріоритезації.

3. Програмна реалізація гібридного алгоритму, включаючи модуль бінарної фільтрації, векторизований розрахунок та логіку жадібного відбору у розподіленій архітектурі.
4. Експериментальне дослідження ефективності розробленої методики та верифікація її адаптивності до зміни зовнішніх динамічних показників.

5. Перелік ілюстративного матеріалу: *презентація*
1. Мета, Об’єкт та Предмет Дослідження.
2. Аналіз Існуючих Підходів та Методів.
3. Математична Модель Динамічної Корисності.
4. Блок-схема Гібридного Алгоритму Комплектації.
5. Архітектура Розподіленої Системи.
6. Демонстрація Збору Динамічних Показників.
7. Результати Моделювання: Сценарний Аналіз.
8. Фінальні Висновки та Наукова Новизна.
9. Публікації та Апробація Роботи.

6. Дата видачі завдання«31» жовтня 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз наявної науково-технічної літератури	31.10.2025	
2	Систематизація динамічних показників та формалізація задачі.	03.11.2025	
3	Обґрунтування вибору гібридного алгоритму та програмних засобів для розподіленої архітектури.	05.11.2025	
4	Програмна реалізація модулів бінарної фільтрації, векторизованого розрахунку та логіки жадібного відбору	11.11.2025	
5	Розробка методики експериментальних досліджень та формування тестових сценаріїв для верифікації адаптивності.	15.11.2025	
6	Проведення експериментальних досліджень та порівняльний аналіз ефективності	16.11.2025	
7	Оформлення роботи: вступ, висновки, реферат	17.11.2025	
8	Розробка демонстраційних матеріалів	18.11.2025	
9	Попередній захист роботи	21.11.2025	
10	Рецензування роботи, перевірка на плагіат	01.12-18.12.2025	
10	Подання роботи до захисту	19.12.2025	

Здобувач вищої освіти _____ Михайло СКВОРЦОВ
(підпис)

Керівник
кваліфікаційної роботи _____ Людмила СОЛЯНИК
(підпис)

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня магістра: 70 стор., 7 табл., 7 рис., 32 джерел.

Мета роботи – автоматизація комплектації багажу для подорожей з урахуванням зовнішніх факторів за рахунок автоматизації вибору речей на основі динамічних показників.

Об'єкт дослідження – процес комплектації багажу для подорожей з урахуванням зовнішніх факторів.

Предмет дослідження – алгоритми та методи оптимізації, що використовуються для автоматизованої комплектації багажу для подорожей.

У роботі проведено аналіз існуючих підходів до комплектації багажу. Проведено класифікацію динамічних показників на ДП-коректори, ДП-обмежувачі та ДП-ризики, що є основою для інтеграції контексту. Розроблено математичну модель «динамічної корисності», яка вводить мультиплікативні коефіцієнти для автоматичної пріоритезації речей. Кінцева корисність розраховується як синтез статичної цінності та динамічних мультиплікаторів, що забезпечує градуальне коригування пріоритетів. Розроблено гібридний алгоритм комплектації багажу, що поєднує жорстку бінарну фільтрацію несумісних предметів із швидким жадібним алгоритмом сортування за розрахованою динамічною корисністю. Програмно реалізовано розроблену методику у форматі розподіленої системи на базі ASP.NET Core (.NET 8) та Blazor Server для клієнтської взаємодії, з використанням Python-мікросервісу для обчислювального ядра. Система інтегрована із зовнішнім API для отримання даних прогнозу погоди в реальному часі. Проведено експериментальне дослідження на основі сценарного моделювання, яке підтвердило перевагу розробленого методу.

КЛЮЧОВІ СЛОВА: ОПТИМІЗАЦІЯ БАГАЖУ, ДИНАМІЧНІ ПОКАЗНИКИ, ГІБРИДНИЙ АЛГОРИТМ, ДИНАМІЧНА КОРИСНІСТЬ, KNAPSACK PROBLEM, СППР, REST API.

ABSTRACT

Text part of the master's qualification work: 70 pages, 7 pictures, 7 table, 32 sources.

The goal of the work is to develop and implement an effective methodology for baggage packing optimization, which will facilitate the formation of a personalized and maximally useful set of items, considering dynamic external factors and strict weight limits.

The object of the study is the baggage packing process for travel, taking into account external factors, namely the analysis and optimization of this process to increase the efficiency and personalization of the item set.

The subject of the study is the adaptive algorithms and optimization methods that allow for the automatic adjustment of item selection and prioritization within weight limits, according to dynamic indicators.

The work analyzes existing approaches to baggage packing. The shortcomings of classic static checklists and dynamic programming algorithms are systematically presented, and their unsuitability for quick recalculation and adaptation under dynamic indicators is substantiated. A classification of Dynamic Indicators into DI-correctors, DI-limiters, and DI-risks has been conducted, forming the basis for context integration.

A mathematical model of "dynamic utility" has been developed. This model, unlike the classical Knapsack Problem, introduces multiplicative coefficients for the automatic prioritization of items. The final utility is calculated as a synthesis of static value and dynamic multipliers, ensuring a gradual adjustment of priorities.

A hybrid baggage packing algorithm has been developed, which combines strict binary filtering of incompatible items with a fast greedy sorting algorithm based on the calculated dynamic utility. This approach ensures a low computational complexity of, which is necessary for operational adaptation.

The developed methodology has been programmatically implemented as a distributed system based on ASP.NET Core (.NET 8) and Blazor Server for client interaction, utilizing a Python microservice for the computational core. The system is

integrated with an external API (OpenWeatherMap) to receive real-time weather forecast data.

An experimental study based on scenario modeling confirmed the advantage of the proposed method. The system demonstrated the ability to adapt the item set, automatically adding specific gear and filtering out irrelevant items, which increases the overall utility of the baggage.

The practical significance of the work lies in the creation of a software system that allows for the automation of the baggage packing process, increasing the efficiency of travel planning and minimizing the risk of forgetting critical items.

KEYWORDS: BAGGAGE OPTIMIZATION, DYNAMIC INDICATORS, DYNAMIC UTILITY, HYBRID ALGORITHM, KNAPSACK PROBLEM, ADAPTIVE DSS, REST API

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

СППР – система підтримки прийняття рішень

КП (KP) – Knapsack Problem (Задача про рюкзак)

ДП – динамічні показники (зовнішні контекстно-залежні змінні)

ІКТ – інформаційно-комунікаційні технології

ORM – Object-Relational Mapper (Об'єктно-реляційний мапер)

EF Core – Entity Framework Core (технологія доступу до даних)

ASP.NET Core – веб-фреймворк для розробки

API Blazor – фреймворк для розробки інтерактивного веб-додатку

JSON – JavaScript Object Notation (текстовий формат обміну даними)

SQL – Structured Query Language (мова структурованих запитів)

REST – Representational State Transfer (архітектурний стиль для розподілених систем)

HTTP – HyperText Transfer Protocol (протокол передачі гіпертексту)

СУБД – система управління базами даних

NP – Non-deterministic Polynomial time (клас складності задач, що не розв'язуються за поліноміальний час)

SIMD – Single Instruction, Multiple Data (архітектура обчислень, що забезпечує векторизацію)

ВСТУП

Актуальність теми дослідження зумовлена стрімким зростанням інтенсивності міжнародного та внутрішнього туризму, що висуває нові вимоги до ефективності та персоналізації процесу підготовки до подорожей. [5;10]

У сучасному світі, де планування поїздок стало динамічним і часто спонтанним, перед кожним мандрівником постає проблема формування оптимального багажу, який не лише відповідатиме суворим обмеженням авіаперевізників щодо ваги та об'єму, але й максимально задовольнятиме потреби в умовах, що можуть змінюватися.

Традиційні підходи до комплектації багажу виявилися неефективними в умовах такої динаміки. Звичайні статичні чеклисти демонструють максимальну простоту реалізації та використання , проте вони є абсолютно негнучкими, не здатні врахувати кількісні обмеження і не адаптуються до унікальних обставин кожної окремої поїздки. Такі списки пропонують універсальний набір, який не гарантує ні корисності, ні дотримання лімітів ваги.

Водночас, класичні методи оптимізації, на кшталт алгоритмів динамічного програмування, хоча й гарантують теоретично глобально оптимальне рішення , вимагають значних обчислювальних ресурсів і є непридатними для швидкого перерахунку та оперативної адаптації при зміні зовнішніх факторів, таких як неочікуваний прогноз погоди чи зміна формату поїздки. У результаті, більшість мандрівників змушені покладатися на інтуїцію або трудомісткі ручні перевірки, що призводить до зайвого багажу або, навпаки, забування критично важливих речей.

Таким чином, у контексті автоматизації планування подорожей виникає гостра потреба у розробці нової інтелектуальної методики, яка б дозволила перейти від статичного вибору речей до автоматизованої комплектації багажу з урахуванням динамічних показників поїздки (як-от погода, тип поїздки, вік, термін поїздки чи стать подорожуючого). Розробка такої методики, яка забезпечить гнучкість, адаптивність та об'єктивну пріоритезацію предметів, є ключовим кроком

до підвищення корисності багажу та оптимізації всього процесу підготовки до подорожі.

Мета роботи – автоматизація комплектації багажу для подорожей з урахуванням зовнішніх факторів за рахунок автоматизації вибору речей на основі динамічних показників. Результату вдається досягти завдяки автоматизованому вибору спорядження, що враховує динамічні параметри та дозволяє замінити статичні списки гнучкою адаптивною системою.

Для досягнення поставленої мети необхідно виконати наступні завдання:

- Дослідження та систематизація впливу зовнішніх динамічних показників на функцію цінності предметів багажу.
- Розробка та математична формалізація моделі динамічної корисності та гібридного алгоритму комплектації багажу на основі адаптивної пріоритезації.
- Програмна реалізація гібридного алгоритму, включаючи модуль бінарної фільтрації, векторизований розрахунок та логіку жадібного відбору у розподіленій архітектурі.
- Експериментальне дослідження ефективності розробленої методики та верифікація її адаптивності до зміни зовнішніх динамічних показників.

Об’єкт дослідження – процес комплектації багажу для подорожей з урахуванням зовнішніх факторів.

Предмет дослідження – алгоритми та методи оптимізації, що використовуються для автоматизованої комплектації багажу для подорожей.

У дослідженні для досягнення поставленої мети було використано комплекс наукових методів. На початковому етапі проведено аналіз літературних джерел та існуючих рішень для вивчення обмежень статичних чеклистів та класичної оптимізації. Подальший етап передбачав застосування методів математичного моделювання та алгоритмізації для розробки моделі динамічної корисності та створення гібридного алгоритму комплектації, який поєднує механізми фільтрації з жадібним алгоритмом сортування.

Наукова новизна одержаних результатів полягає у:

1. Вперше розроблено математичну модель «динамічної корисності», яка, на відміну від класичної задачі про рюкзак, вводить мультиплікативні коефіцієнти (погода, тип подорожі, вік, стать) для автоматичної пріоритезації речей.

2. Розроблено гібридний алгоритм комплектації багажу, який поєднує фільтрацію несумісних предметів із жадібним сортуванням за розрахованою корисністю, що забезпечило подальший розвиток підходів до оптимізації багажу

3. Удосконалено підхід до оптимізації багажу, що дозволяє системі автоматично адаптувати набір речей під зовнішні умови, додаючи специфічне спорядження та відсіюючи нерелевантне, що підвищує загальну корисність багажу.

Практичне значення одержаних результатів полягає в можливості впровадження розробленої програмної системи, яка дозволить автоматизувати процес комплектації багажу для подорожей

Апробація результатів магістерської роботи. Основні положення та результати дослідження доповідалися та обговорювалися на науково-практичних конференціях та семінарах з питань інформаційних технологій в освіті.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ІСНУЮЧИХ ПІДХОДІВ ДО ОПТИМІЗАЦІЇ

1.1. Аналіз предметної області та теоретичних основ

Задача оптимізації комплектації багажу для подорожей є класичною прикладною задачею, що в своїй основі належить до класу задач про рюкзак. [2,11] Ця проблема набуває особливої актуальності у контексті сучасних систем підтримки прийняття рішень, де її розв'язання вимагає вибору обмеженого набору предметів, які максимізують певний критерій корисності, не перевищуючи жорстко задані обмеження, головним чином, за вагою. [6]

Knapsack Problem – це NP-складна задача комбінаторної оптимізації, в якій необхідно обрати найбільш цінну комбінацію предметів (наш багаж), щоб їхня сумарна вага не перевищувала максимальну допустиму місткість «рюкзака». У традиційній формі КР спрямована на вибір набору предметів із максимальною цінністю при дотриманні обмеження на вагу.

Для цілей комплектації багажу ця модель виходить за межі простої КР. По-перше, вона потребує багатовимірності лише за параметрами, що впливають на якість рішення (наприклад, вага та кількість предметів), оскільки фізичні обмеження можуть бути виключені. По-друге, критично важливою є багатокритеріальність. На відміну від стандартної КР, де оптимізується лише цінність, тут необхідно одночасно враховувати та балансувати кілька цільових функцій: максимізувати корисність, мінімізувати вартість (ціну купівлі забутих речей) і зменшувати надлишковість(кількість непотрібних речей), що особливо актуально для подорожей з різними цілями. [3]

Розв'язання цієї задачі є ключовим для подорожей із різними цілями (відрядження, відпочинок, спортивні змагання), оскільки:

1. Цінність одного й того самого предмета кардинально змінюється залежно від мети. Наприклад, ноутбук має високу цінність для ділової поїздки, але низьку для пляжного відпочинку.

2. Обмеження за вагою може бути жорстко регламентовано авіакомпаніями та змінюватися залежно від класу квитка чи типу багажу (ручна поклажа чи зареєстрований).

3. Неправильна комплектація призводить до фінансових втрат (плата за перевищення ваги, купівля забутих речей) або зниження якості самої подорожі (необхідність носити надмірну вагу).

Ключовим недоліком класичних оптимізаційних моделей, які є статичними, є їхня неспроможність адекватно реагувати на зовнішні чинники. Параметри предметів (їхня вага, цінність) та обмеження (максимальна вага) фіксовані, але реальний процес комплектації підпорядковується зовнішнім чинникам, які є динамічними – вони змінюються в часі або залежать від поточного контексту.

Динамічні показники – це зовнішні, контекстно-залежні змінні, які впливають на значення корисності окремого предмета або на жорсткість вагових обмежень у процесі оптимізації комплектації. Саме інтеграція цих показників є основою розроблюваної методики.

Таблиця 1.1

Класифікація та вплив динамічних показників

Категорія ДП	**Приклади динамічних показників**	**Вплив на параметри моделі**
Гео-часові	Прогноз погоди (температура, опади, сонячна активність), тривалість/сезонність поїздки.	Динамічна зміна корисності одягу та аксесуарів (наприклад, цінність теплого светра зростає при зниженні температури).

Користувацькі	Потреби, що змінюються (плановані екскурсії, наявність сервісів, поточний пріоритет).	Зміна корисності (наприклад, цінність купальника, якщо подорож користувача виключно ділова).
Логістичні	Фактична вага предметів, поточні обмеження авіакомпаній (які можуть змінюватися після бронювання).	Зміна доступного ліміту ваги в реальному часі.

Інтеграція ДП дозволяє перейти від статичного рішення до адаптивної системи, яка пропонує оптимальний набір, що є релевантним саме для цієї поїздки та поточного контексту. Методика має забезпечити математичний апарат, здатний перетворити ці зовнішні, часто якісні або стохастичні показники на кількісні параметри цінності та обмежень для моделі Knapsack. Це є ключовим викликом, який буде розв'язано у подальших розділах роботи.

1.2. Огляд та порівняльний аналіз існуючих підходів

Дані, що стосуються предметів багажу та умов подорожі, є специфічним класом інформаційних ресурсів, властивості яких кардинально відрізняються від статичних наборів, використовуваних у традиційних задачах. Розуміння цієї специфіки є фундаментальним для побудови ефективної методики оптимізації комплектації.

Для розв'язання задачі Knapsack, яка є основою для предметної області, історично застосовувалися декілька основних класів алгоритмів. Класичні точні методи, такі як динамічне програмування та метод гілок і меж, були розроблені для гарантованого знаходження глобально оптимального рішення. [11] Динамічне програмування ефективне для одновимірної КР, оскільки воно базується на

поступовому розрахунку оптимальних рішень для підзадач. Проте його застосування різко обмежується, коли максимальна вага або кількість предметів стає великою. Для реальних СППР, де важлива швидкість, ці методи часто є неприйнятними.

З огляду на те, що комплектація багажу є складною багатокритеріальною задачею, де точне оптимальне рішення не може бути отримано за розумний час, широкого поширення набули евристичні та метаевристичні підходи. Вони дозволяють знайти якісне, хоча й не гарантовано оптимальне, рішення за значно менший час. До них відносяться:

- Жадібні алгоритми - вирізняються простотою та високою швидкістю, оскільки обирають предмети за локальним критерієм (наприклад, найкращим співвідношенням цінності до ваги). Попри швидкість, їхнім головним недоліком є схильність до субоптимальних рішень, що неприпустимо при пакуванні критично важливого багажу.
- Генетичні алгоритми – імітують природний відбір і є дуже ефективними для пошуку якісних рішень у великих просторах. Вони легко адаптуються до багатокритеріальної оптимізації, що дозволяє одночасно максимізувати корисність речей та мінімізувати їхню вагу, хоча це часто вимагає значних обчислювальних ресурсів та часу для збіжності алгоритму. [4,7]
- Нейронні мережі та методи імітації відпалу – також застосовуються для складних NP-задач, проте вимагають значних обчислювальних ресурсів для навчання. Часто їхня складність не виправдана, якщо проблему можна вирішити менш «важкими» методами. [3]

Часто їхня складність не виправдана, якщо проблему можна вирішити менш «важкими» методами. Для систематизації розглянутих підходів до розв'язання задачі комплектації багажу побудовано ієрархічну класифікацію, наведену на рисунку 1.1

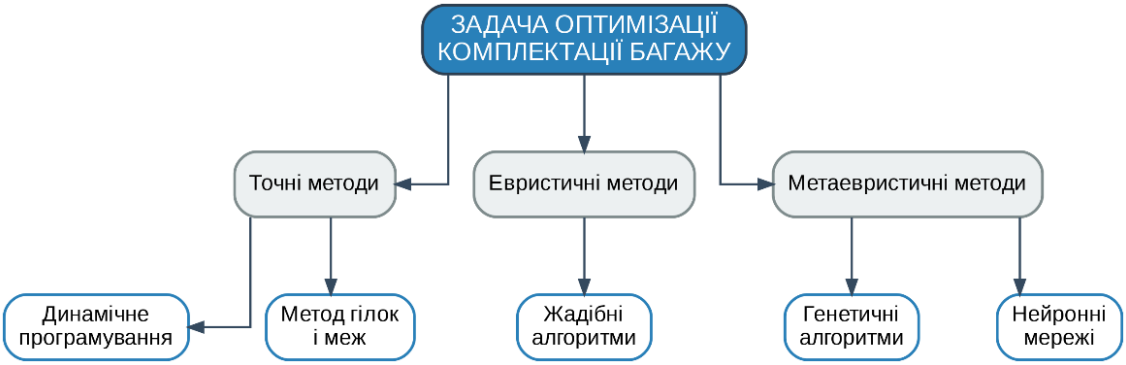


Рис. 1.1 Ієрархічна класифікація підходів розв'язання задачі комплектації багажу

Незважаючи на різноманіття методів, представлених на схемі, ключовим недоліком більшості з них залишається орієнтація на статичні дані. Хоча вони можуть впоратися зі складністю та багатовимірністю, їхнє застосування не вирішує проблему інтеграції динамічних показників. ДП залишаються зовнішнім фактором, який необхідно спочатку правильно перетворити в статичні параметри цінності та обмежень, перш ніж алгоритм почне працювати.

Для систематизації та візуалізації недоліків існуючих підходів, які застосовуються в СППР для пакування, проведено їх порівняльний аналіз, представлений у Таблиці 1.2. Порівняння охоплює три основні групи рішень: статичні чеклисти, класичні алгоритми оптимізації та декларативні системи на основі правил.

Таблиця 1.2

Аналіз існуючих підходів та методів

Метод	**Використовуваний алгоритм**	**Час Виконання**	**Переваги**	**Недоліки**
Статичні чеклисти	Пошук у таблиці	O(1) або O(N)	Максимальна швидкість. Простота реалізації та використання.	Рішення не враховує кількісні обмеження (вага/об'єм).

Класична оптимізація	Алгоритми Динамічного Програмування	O(nW)	Гарантує глобально оптимальне рішення	Непридатність для швидкого перерахунку при зміні умов.
Декларативні системи	Система правил	O(N)	Чітке моделювання бінарних обмежень. Можливість врахувати мінімальну логіку.	Не може присвоювати плаваючі вагові коефіцієнти динамічним показникам.

Більшість існуючих комерційних мобільних додатків та онлайн-сервісів для пакування реалізують функціонал, що відповідає методам, описаним у Таблиці 1.2: статичні чеклисти (на основі шаблонів) або декларативні системи (мінімальна логіка "якщо дощ, додати парасольку"). Вони забезпечують певну зручність, але не є справжніми оптимізаційними системами.

Проведений аналіз виявив три ключові системні недоліки існуючих прикладних рішень, які безпосередньо випливають із їхньої неспроможності ефективно інтегрувати динамічні показники:

1. Найпросунутіші СППР використовують ДП (наприклад, прогноз погоди) лише як бінарний фільтр, а не як фактор оптимізації. Це означає, що предмет або додається (1), або виключається (0). Однак система не може динамічно перерахувати цінність предмета в залежності від ступеня важливості – наприклад, цінність дощовика при 90% ймовірності опадів повинна бути значно вищою, ніж при 30%. Існуючі рішення не можуть присвоювати плаваючі вагові коефіцієнти цінності, що призводить до субоптимального набору багажу.

2. Оскільки подорож має динамічний характер (зміна маршруту, оновлення прогнозу погоди), СППР повинні мати можливість швидко перерахувати оптимальний склад. Використання точних, але повільних алгоритмів (динамічного

програмування) стає неможливим. Системи, які не можуть оперативно адаптуватися до змін, втрачають свою цінність для користувача.

3. Прикладні рішення ігнорують складну цільову функцію, необхідну для подорожей з різними цілями. Наприклад, для ділової поїздки система має максимізувати корисність ділового одягу та мінімізувати ризик забути презентаційне обладнання, тоді як для відпочинку – мінімізувати загальну вагу та кількість непотрібних речей. Існуючі СППР не пропонують механізмів для інтеграції цих змінних пріоритетів.

Проведений аналіз підтверджує, що існуючі наукові підходи та прикладні рішення не мають ефективного універсального механізму для інтеграції динамічних показників безпосередньо у функцію цінності та вагових обмежень моделі Knapsack.

Це формує наукову проблему, яка полягає у необхідності розробки методики, здатної забезпечити:

- Розробку механізму математичного перетворення зовнішніх, контекстних та імовірнісних ДП (наприклад, "висока ймовірність дощу") у кількісні, плаваючі параметри цінності та обмежень для оптимізаційної моделі.
- Створення моделі, яка може динамічно перерахувати оптимальний склад багажу, балансуючи між користю, вагою та ризиками, відповідно до змін зовнішнього контексту та цілей подорожі.

Саме розробка такої адаптивної та динамічної методики оптимізації комплектації багажу є основним завданням цієї магістерської роботи.

1.3. Аналіз структури динамічних показників

Низький рівень достовірності результатів, отриманих за допомогою існуючих прикладних рішень, є прямим наслідком їхньої нездатності працювати з динамічними показниками, як з імовірнісними, контекстно-залежними величинами. Для розробки ефективної методики оптимізації необхідно провести

детальну декомпозицію структури ДП та визначити механізми їхнього впливу на елементи математичної моделі.

ДП є не ізольованими параметрами, а формують складний контекст, який визначає функцію цінності кожного предмета багажу. У контексті роботи, ми класифікуємо ДП на три типи відповідно до їхньої дії на модель Knapsack:

- ДП-коректори - це зовнішні показники (наприклад, прогноз погоди, ціль поїздки), які змінюють цінність предмета, впливаючи на коефіцієнт корисності, але не змінюючи його вагу. Наприклад, якщо температура різко знизилася, цінність теплого светра має зрости не бінарно (додати/виключити), а пропорційно. Цей клас ДП вимагає перетворення якісних або числових значень на плаваючі вагові коефіцієнти цінності.
- ДП-обмежувачі - це показники, які змінюють жорсткість вагового обмеження у режимі реального часу. Прикладом є оновлення інформації про ліміт ручної поклажі авіакомпанії, або рішення користувача про необхідність залишити певний резерв ваги для подальших покупок. Ці показники вимагають розробки адаптивного механізму перерахунку доступної ваги.
- ДП-ризики - це показники, що вводять імовірнісну складову (наприклад, ймовірність дощу, ризик медичної потреби). Ці ДП вимагають розробки механізму формалізації невизначеності шляхом присвоєння предмету цінності, що базується на ймовірності його необхідності та оцінці ціни втрат від його відсутності.

Для узагальнення наведеної класифікації та наочної демонстрації того, як саме ці різнорідні показники трансформуються у параметри математичної моделі, розроблено структурну схему впливу, що зображена на рисунку 1.2

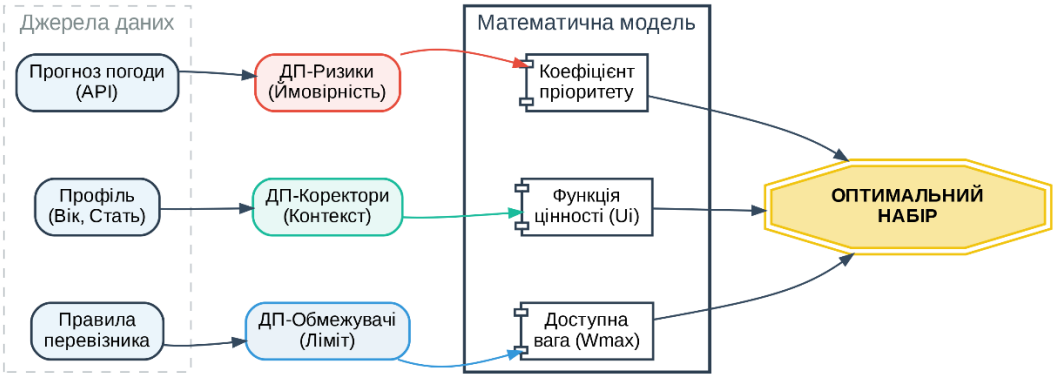


Рис. 1.2 Схема впливу

Як показано на схемі, структурна декомпозиція дозволяє чітко розмежувати сфери впливу кожного показника: ризики визначають пріоритетність, контекст коригує цінність, а обмежувачі встановлюють фізичні ліміти. Така організація вхідних даних є основою для побудови математичної моделі, яка буде розглянута в наступному розділі.

Проблема полягає в тому, що існуючі методики не надають адекватного математичного апарату для перетворення цих різнорідних ДП у числові параметри оптимізаційної моделі (наприклад, функції належності або динамічних вагових коефіцієнтів). Ця декомпозиція та подальша розробка механізмів формалізації кожного типу ДП є ключовим невирішеним аспектом, який лягає в основу наукової новизни цієї роботи.

Таким чином, проведений аналіз показав, що існуючі прості методи не здатні комплексно вирішити проблему адаптивної комплектації багажу в умовах динамічних показників. Це обґрунтовує необхідність розробки методики оптимізації комплектації багажу на основі динамічних показників, яка поєднувала б механізми формалізації ДП для перерахунку цінності та багатокритеріального алгоритму для ухвалення рішення в умовах жорстких вагових обмежень.

2 РОЗРОБКА МЕТОДИКИ ТА МАТЕМАТИЧНОЇ МОДЕЛІ АДАПТИВНОЇ ОПТИМІЗАЦІЇ КОМПЛЕКТАЦІЇ БАГАЖУ

2.1. Аналіз методів оптимізації та етапи методики

2.1.1 Критичний аналіз існуючих підходів до розв'язання задачі комплектації багажу

Для розробки ефективної методики оптимізації комплектації багажу необхідно критично оцінити обчислювальну складність алгоритмічних рішень, оскільки пропоноване рішення має функціонувати як засіб підтримки прийняття рішень у режимі динамічної зміни вхідних даних.

Класичні точні методи, такі як алгоритми динамічного програмування, є неприйнятними для задач з високою швидкістю зміни контексту. Хоча ДП гарантує знаходження глобального оптимуму для задачі Knapsack, його обчислювальна складність є псевдополіноміальною – $O(nW)$ (де n – кількість предметів, а W – максимальна вага). При значних вагових обмеженнях та великій кількості потенційних предметів, час виконання стає надмірним, що унеможливлює оперативну адаптацію системи до змін динамічних показників. [3,11]

З огляду на необхідність миттєвої реакції СППР, основою методики було обрано жадібний алгоритм. Його ключова перевага полягає у низькій обчислювальній складності $O(NlogN)$, яка визначається лише часом на сортування предметів за ключовим критерієм. [3] Ця швидкість дозволяє системі оперативно перерахувати рішення при кожній зміні ДП.

Для подолання головного недоліку жадібного алгоритму – схильності до локального оптимуму – розроблена методика, що використовує гібридну архітектуру. Така архітектура реалізована через попередню адаптивну пріоритезацію предметів, де цінність кожного елемента обчислюється на основі всіх динамічних показників. [14] Ключовим критерієм для сортування, відповідно

до розробленого алгоритму, є динамічна корисність $U(i,P)$. Ця метрика замінює статичне оцінювання цінності предмета на контекстно-залежний показник, що підвищує якість рішення. Об'єктом оптимізації є сумарна корисність комплекту U_{total} (2.1):

$$U_{total} = \sum_{\{i \in X\}} U(i,P) \rightarrow max\,, \qquad (2.1)$$

при обмеженні (2.2) :

$$\sum_{\{i \in X\}} w_i \leq W_{max}\ , \qquad (2.2)$$

де U_{total} – сумарна корисність усього комплекту багажу.

X – множина (комплект) речей, обраних для багажу.

i – окрема річ, що розглядається.

$U(i,P)$ – динамічна корисність речі *i* у контексті *P*.

P – динамічний контекст поїздки (набір показників: погода, тип поїздки, супутники).

w_i – вага речі *i*.

W_{max} – максимальна допустима вага багажу.

Таким чином, у розробленій методиці жадібний алгоритм базується на сортуванні предметів за абсолютним значенням $U(i,P)$, оскільки вага предмета враховується на наступному етапі, що забезпечує швидкість та адаптивність системи. [13]

2.1.2. Етапи та послідовність дій методики адаптивної оптимізації

Запропонована методика є гібридним адаптивним рішенням, розробленим для подолання обмежень статичних чеклистів та обчислювальної складності класичних алгоритмів динамічного програмування. Методика використовує інтелектуальний підхід, поєднуючи механізми попередньої бінарної фільтрації несумісних предметів з подальшим застосуванням швидкого жадібного алгоритму.[4] Ключова відмінність розробленого гібридного алгоритму полягає в

інтеграції математичної моделі динамічної корисності $U(i,P)$, а також використанні абсолютного критерію сортування жадібного алгоритму за значенням $U(i,P)$. Це дозволяє системі оперативно формувати ранжований список предметів, де найвищий пріоритет мають ті, чия цінність була найбільше адаптована до поточних зовнішніх умов (погоди, віку тощо). Таким чином, корисність визначає порядок включення, а вага – фізичну можливість включення на наступному кроці. [3]

Упровадження комплексної адаптивної логіки вимагає чітко структурованої послідовності кроків. Від моменту збору вихідних даних до фінальної рекомендації комплекту багажу, процес виконується у декілька послідовних фаз. Кожен етап цієї процедури має критичне значення для коректної трансформації якісних динамічних показників у кількісні параметри оптимізаційної моделі.

Загальна послідовність дій, реалізована в рамках розробленої методики, представлена на блок-схемі алгоритму комплектації багажу на основі динамічних показників на рисунку 2.1.

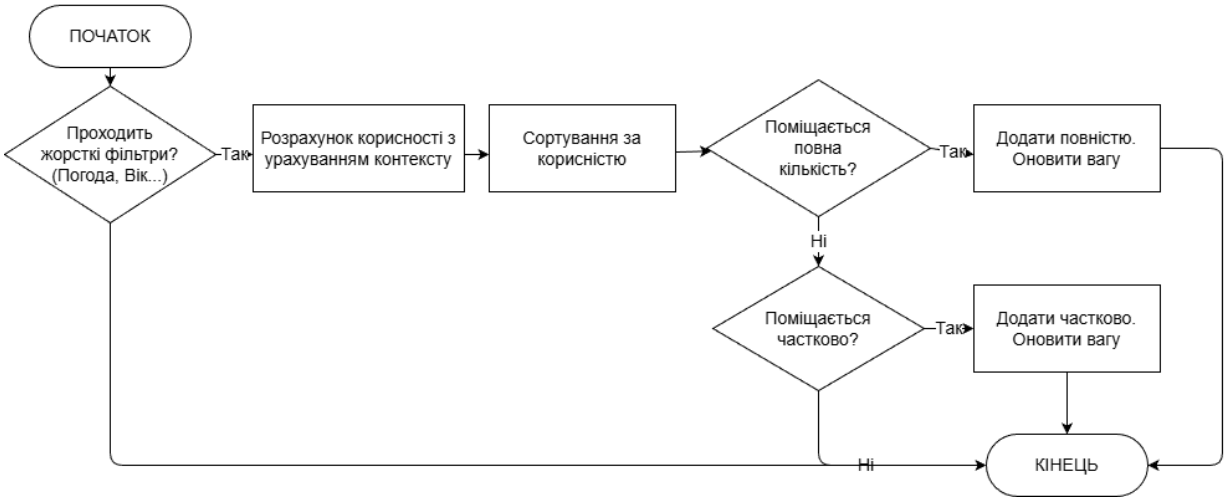


Рис 2.1 – Блок-схема алгоритму комплектації багажу

Розглянемо детальніше кожен із ключових етапів, що забезпечують функціонування даного алгоритмічного рішення.

Етап 1. Формалізація вхідних даних та визначення динамічного контексту

На цьому первинному етапі забезпечується агрегація та стандартизація всього масиву вхідної інформації, необхідної для функціонування оптимізаційної моделі. Основна мета полягає у формуванні повноцінного динамічного контексту поїздки, який слугуватиме основою для адаптивної пріоритезації.

Критично важливим кроком є збір та інтеграція динамічних показників – зовнішніх параметрів, які впливають на корегування цінності кожного предмета в інвентарі. Ці показники включають як параметри, задані користувачем (наприклад, тип подорожі, вік, стать, тривалість поїздки, тощо), що формують первинний профіль потреб, так і критично важливі дані (ДП-коректори), отримані у режимі реального часу через інфокомунікаційні технології. До останніх належить, зокрема, прогноз погоди (температура, рівень опадів, сонячна активність), який має безпосередній вплив на функціональність та необхідність предметів. Інтеграція із зовнішніми API (наприклад, OpenWeatherMap) дозволяє отримувати ці ДП-коректори у режимі реального часу, що є ключовим фактором забезпечення адаптивності та релевантності оптимізаційного рішення.

Паралельно цьому процесу фіксується максимальна допустима вага багажу. Важливо підкреслити, що це обмеження може мати гнучкий характер, забезпечуючи можливість зміни його жорсткості у режимі реального часу. Це може бути спричинено як оновленням нормативних правил авіакомпаній чи перевізників, так і рішенням користувача про залишення резерву ваги для придбань або непередбачених обставин.

Також, на цьому етапі відбувається формалізація часових параметрів та розрахунок необхідної кількості. Параметр тривалості подорожі $T_{\text{дні}}$ є фундаментальним для обчислення базової необхідної кількості предметів, особливо для тих, що використовуються щоденно. Визначення $Q_{\text{необх}}$ є критично важливим для уникнення надлишкового обсягу багажу, який неминуче призводить до перевищення вагових лімітів. Для забезпечення інтелектуальної оптимізації та мінімізації надмірності, вводиться параметр періодичності використання $D_{\text{період}}$.

Цей параметр відображає логістичні властивості предмета та його здатність до багаторазового використання або прання в умовах подорожі. Використання $D_{\text{період}}$ дозволяє системі розрахувати раціонально мінімальну кількість $Q_{\text{необх}}$, яка забезпечить потреби мандрівника протягом усієї поїздки, але при цьому запобігатиме нераціональному пакуванню. Це досягається шляхом масштабування тривалості поїздки відносно періоду використання предмета. Таким чином, розрахунок мінімально необхідної кількості $Q_{\text{необх}}$ базується на відношенні тривалості подорожі до періодичності використання, з обов'язковим врахуванням обмеження на максимальну допустиму кількість Q_{max} (наприклад, для уникнення пакування 100 одиниць предмета). Розрахунок необхідної кількості $Q_{\text{необх}}$ формалізується із застосуванням функції заокруглення вгору(2.3):

$$Q_{\text{необх}} = \min\left(Q_{\text{max}}, \left\lceil \frac{T_{\text{дні}}}{D_{\text{період}}} \right\rceil\right), \qquad (2.3)$$

де $Q_{\text{необх}}$ – необхідна кількість одиниць предметів для пакування;

Q_{max} – максимально допустима кількість одиниць предмета;

$T_{\text{дні}}$ – тривалість поїздки, у днях;

$D_{\text{період}}$– періодичність використання предмета (кількість днів між використаннями). [2]

Далі підготовлюється масив даних для обробки. Всі статичні дані про потенційні предмети (базова корисність, вага, категорія, максимальна кількість) витягуються з бази даних. І цей масив даних, разом із розрахованою необхідною кількістю та динамічним контекстом, є основою для подальших етапів фільтрації та оптимізації.

Етап 2. Жорстка бінарна фільтрація несумісних предметів.

Цей етап являє собою первинний механізм скорочення простору пошуку, який базується на бінарній логіці. Його основне завдання – підвищити обчислювальну ефективність алгоритму, виключаючи необхідність обробляти очевидно нерелевантні предмети, чия корисність у заданому контексті дорівнює нулю. Це є критичним для оперативності СППР. [12]

Принцип функціонування базується на використанні виключно бінарної логіки. На відміну від плаваючого та мультиплікативного коригування цінності на наступному етапі, тут перевіряється лише факт абсолютної несумісності. Якщо предмет не відповідає жорсткій умові, його корисність прирівнюється до нуля, і він негайно виключається з оптимізаційного циклу, що забезпечує радикальне скорочення множини N.

Декомпозиція фільтраційних правил передбачає аналіз найбільш критичних ДП-коректорів $\boldsymbol{P}_{\text{фільтр}}$ Ці правила застосовуються послідовно та детально описані у Таблиці 2.1.

Таблиця 2.1

Класифікація правил жорсткої бінарної фільтрації

Категорія фільтрації	**Обґрунтування застосування (ДП-коректор)**	**Принцип бінарного виключення**
Гендерна та вікова відповідність	Забезпечує персоналізацію на базовому рівні, гарантуючи, що речі відповідають фізіологічним та віковим потребам мандрівника	Виключення предмета, якщо його правило не збігається з віковими групами або статтю мандрівника.
Тип поїздки та її мета	Усуває спорядження, яке є нерелевантним для загальної мети подорожі.	Виключення вузькоспеціалізованих речей, якщо їхній критичний тип поїздки не збігається з контекстом.

Таким чином, успішне проходження жорсткої бінарної фільтрації є обов'язковою умовою для переходу предмета на наступний етап, де вже відбувається плаваючий розрахунок корисності.

Етап 3. Формалізація та отримання динамічного коректора $\boldsymbol{P}_{\text{погода}}$

Цей етап присвячений інтеграції зовнішніх, змінних даних (прогнозу погоди) у систему і є логічною частиною загальної стратегії адаптації. Його основне завдання – обчислити коефіцієнт градуального коригування ($P_{\text{погода}}$), який буде

використаний на наступному кроці для динамічного зважування цінності предметів. На відміну від бінарних фільтрів, цей коефіцієнт є справжнім динамічним мультиплікатором, що належить до діапазону [0; 2].

Обчислення коефіцієнта $P_{\text{погода}}$ вимагає надійної та оперативної інтеграції з зовнішніми джерелами даних. Самі дані про ймовірність опадів отримуються в реальному часі через зовнішній сервіс, OpenWeatherMap API. Обробка даних здійснюється мікросервісом Python, який викликає зовнішній сервіс, нормалізує отримані дані та повертає необхідні числові показники для розрахунку $P_{\text{погода}}$. Це підкреслює залежність системи від ІКТ-інтеграції.

Розрахунок цього коректора має дві фази, залежно від ймовірності опадів. Перша фаза – градуальне коригування ($P_{\text{погода}} \in [0; 1]$). Якщо ймовірність опадів є низькою або помірною, динамічний коректор розраховується пропорційно до ймовірності, що дозволяє плавно знижувати загальну корисність погодних предметів.

Друга фаза – динамічне підсилення ($P_{\text{погода}} \in [1; 2]$). Якщо ймовірність опадів перевищує встановлений пороговий рівень (наприклад, 50%), використовується спеціальна формула підсилення. Це гарантує, що при високому ризику, предмети, критичні для виживання або комфорту, отримають пропорційно вищий бонус, підвищуючи їхню корисність до максимуму 2. Критичність динамічного оновлення полягає у тому, що використання даних, отриманих через API, робить показники $P_{\text{погода}}$ мінливими у часі. Чутливість моделі до градації є критичною: сама можливість плавного коригування залежить від отримання числової ймовірності опадів. Якщо зовнішній сервіс не надає ймовірності, градуальний розрахунок стає неможливим, і система змушена переходити до найближчого бінарного рішення. Успішне завершення цього етапу надає системі ключовий динамічний фактор, необхідний для остаточного розрахунку корисності.

Етап 4. Розрахунок динамічної корисності.

Цей етап є завершальним кроком у формуванні критерію сортування, який безпосередньо визначатиме порядок вибору предметів у жадібному алгоритмі. Тут відбувається мультиплікативний синтез статично зваженої основи та градуального мультиплікатора ($P_{\text{погода}}$) отриманого на 3 етапі, для отримання фінального показника $U(i, P)$.

Далі розглянемо фінальну формулу розрахунку та її інтелектуальне зважування. Необхідно підкреслити, що ця формула застосовується виключно до предметів, які не мають абсолютного пріоритету (тобто не є $S_{\text{закр}}$ або базовими речами). Ці предмети вже отримали максимальне значення $U(i, P)$ на попередніх етапах і гарантовано включаються до багажу, якщо дозволяє ваговий ліміт. Для всіх інших предметів, які успішно пройшли жорстку бінарну фільтрацію (етап 2), кінцева динамічна корисність розраховується за формулою, яка інтегрує всі динамічні показники

$$U_{i,P} = (S_{\text{баз}} + S_{\text{вага}}) \times P_{\text{погода}} \times \prod_{j \in \text{фільтр}} P_j \,, \qquad (2.4)$$

де $U_{i,P}$ – кінцева динамічна корисність предмета i у контексті P;

$S_{\text{баз}}$ – базова статична зважена цінність предмета;

$S_{\text{вага}}$ – статична цінність предмета, обумовлена його вагою;

$P_{\text{погода}}$ – коефіцієнт, що враховує вплив прогнозу погоди;

P_j – бінарний динамічний фільтр (коефіцієнт, що набуває значень 0 або 1);

$j \in \text{фільтр}$ – множина бінарних фільтрів, застосованих до предмета.

Ця формула виконує функцію інтелектуального зважування. Вона явно показує, що кінцева корисність предмета залежить як від його внутрішньої, статично зваженої цінності $(S_{\text{баз}} + S_{\text{вага}})$, так і від добутку всіх динамічних коефіцієнтів. Члени P_j (тобто $P_{\text{стать}}\, P_{\text{вік}}\, P_{\text{тип поїздки}}$) у цьому добутку належать до множини $\{0; 1\}$. Це підтверджує, що якщо хоча б один бінарний фільтр не пройдено (коефіцієнт дорівнює нулю), то вся динамічна корисність предмета також обнуляється, незалежно від його базової цінності чи погодного коефіцієнта.

Операційно, в алгоритмі, ця формула реалізується послідовно: спочатку відбувається жорстка бінарна фільтрація, де перевіряється добуток коефіцієнтів $P_{\text{погода}}, P_{\text{вік}}$ та $P_{\text{тип}}$. Якщо цей добуток дорівнює нулю, подальший розрахунок не потрібен. Якщо ж добуток дорівнює одиниці, тоді значення коригується градуальним мультиплікатором $P_{\text{погода}}$, що забезпечує необхідне динамічне підсилення або зниження пріоритету. Отримане значення стає єдиним числовим мірилом відносної важливості предмета в поточному контексті подорожі.

Для забезпечення прозорості та деталізації процесу розрахунку динамічної корисності, нижче наведено декомпозицію ключових компонентів, їхні діапазони значень та функціональне призначення в рамках розробленої моделі. Таблиця 2.2 візуалізує, як саме відбувається адаптивне зважування предметів перед застосуванням жадібного алгоритму.

Таблиця 2.2

Декомпозиція ключових компонентів

Компонент	**Тип**	**Діапазон значень**	**Функція в моделі**
$\boldsymbol{S}_{\text{баз}}$	Статична основа	[100; 500]	Визначає первинну ієрархію цінності предмета, відображаючи його функціональну значущість для будь-якої поїздки
$\boldsymbol{S}_{\text{вага}}$	Гібридний корелятор	10% від $\boldsymbol{S}_{\text{баз}}$ (якщо $w_i \leq$ 300 г)	Надає бонус за компактність і легкість. Стимулює включення легких предметів.
$\boldsymbol{P_j}$	Бінарні фільтри	{0;1}	Забезпечує жорстке виключення (фільтрацію) предметів, якщо вони не відповідають ключовим бінарним критеріям (стать, вік, тип поїздки)
$\boldsymbol{P}_{\text{погода}}$	Градуальний мультиплікатор	[0; 2]	Здійснює адаптивне коригування цінності залежно від зовнішніх

			динамічних факторів, таких як ймовірність опадів.

Етап 5. Сортування та застосування жадібного алгоритму.

Після того, як для кожного предмета була обчислена динамічна корисність $U(i,P)$ наступає ключовий етап безпосередньої оптимізації та формування фінального комплекту багажу. Цей процес забезпечує розв'язання задачі Knapsack, інтегрованої в загальну методику.

Першим кроком є ранжування кандидатів. Усі предмети, що пройшли попередні етапи, сортуються у порядку спадання їхньої динамічної корисності. Обґрунтування вибору жадібного алгоритму полягає у необхідності досягнення високої оперативності системи, оскільки її обчислювальна складність визначається лише часом, необхідним для сортування кількості предметів. Така швидкість є критично важливою для систем підтримки прийняття рішень, що функціонують в умовах динамічної зміни контексту. Попередня інтелектуальна пріоритезація предметів за значенням $U(i,P)$ ефективно мінімізує недоліки жадібного підходу, підвищуючи якість рішення до рівня, що відповідає практичним вимогам.

Після ранжування відбувається ітеративне наповнення багажу, яке суворо контролюється ДП-обмежувачем – максимально допустимою вагою. Система послідовно обирає предмети з початку відсортованого списку, намагаючись максимізувати загальну корисність.

На кожній ітерації алгоритм здійснює дворівневу перевірку. Спочатку він визначає необхідну кількість предмета, яка була розрахована на першому етапі з урахуванням тривалості поїздки. Далі перевіряється умова: чи поміщається сумарна вага у доступний ліміт ваги.

Якщо предмет поміщається повністю, він додається до фінального комплекту, а доступна вага оновлюється. У випадку, коли повна необхідна кількість предмета перевищує залишок ліміту, алгоритм демонструє свою гнучкість: він перевіряє можливість його додавання частково. Це реалізується шляхом зменшення $Q_{необх}$ до максимальної кількості, яка фізично поміститься у

залишок максимальної ваги, що забезпечує максимальне використання доступного вагового ліміту для найбільш пріоритетних речей.

Процес ітерації завершується, коли або вичерпано доступний ліміт W_{max}, або пройдено всі предмети, корисність яких більша за нуль. Таким чином, сформований комплект є оптимальним рішенням, яке максимізує сумарну динамічну корисність при жорсткому дотриманні вагового обмеження $\sum w_i \leq W_{max}$.

Етап 6. Фіналізація та формування результату.

На заключному етапі система здійснює фінальну обробку та комунікацію результату, що є необхідним елементом взаємодії системи підтримки прийняття рішень з користувачем. Після завершення роботи жадібного алгоритму формується фінальна множина, яка є оптимальним комплектом обраних речей, що максимізує цільову функцію при дотриманні всіх вагових та логістичних обмежень. Фінальний список рекомендованих речей з їхньою кількістю та загальною вагою передається користувачеві, забезпечуючи швидке і зручне ухвалення рішення.

Таким чином, гібридна методика забезпечує ефективну та адаптивну оптимізацію комплектації, поєднуючи радикальне скорочення простору пошуку (бінарна фільтрація), адаптивну пріоритезацію (модель $U_{\text{дин}}$) та швидку оптимізацію (жадібний алгоритм). Це дозволяє трансформувати якісні зовнішні фактори у кількісний критерій, максимізуючи використання вагового ліміту для найбільш необхідних речей у поточному контексті.

2.2. Математичне моделювання ключових компонентів методики

Даний пункт присвячений детальному формалізованому опису математичних моделей, що лежать в основі гібридної методики, зокрема структурі моделі динамічної корисності $U_{\text{дин}}$ та механізмам формування логістичних обмежень.

2.2.1. Структура та декомпозиція моделі динамічної корисності

Перехід від статичної оцінки цінності предметів до динамічної, контекстно-залежної корисності є ключовим елементом наукової новизни. Формалізація цієї функції ґрунтується на принципі гібридного коригування, що дозволяє системі градуально адаптувати пріоритети речей до актуального контексту подорожі *(P)*. Ця модель чітко розділяє статичну основу, модифіковану бонусами, від адаптивних мультиплікаторів.[1]

Насамперед, у системі діє абсолютний пріоритет ($S_{\text{зак}}$), який обчислюється першим і є найвищим рівнем пріоритету. Цей компонент присвоюється предметам, які користувач свідомо позначив як обов'язкові, або які є життєво необхідними. Введення абсолютного динамічного бонусу (500% від $S_{\text{баз}}$) функціонує як захисний механізм. Надання цим елементам максимально можливого числового значення гарантує, що вони отримають абсолютну перевагу в процесі сортування, незалежно від низької $S_{\text{баз}}$ або негативного впливу $P_{\text{дин}}$. Це є критичним для забезпечення надійності системи: алгоритм може оптимізувати вагу, але не повинен виключати критичні предмети.

Також, предмети, визначені як базові (наприклад, засоби гігієни), також отримують високий пріоритет, що розраховується як 300% від $S_{\text{баз}}$. Це гарантує, що основні функціональні предмети будуть включені до багажу.

Для незакріплених та небазових предметів встановлюється статична основа, представлена сумою ($S_{\text{баз}} + S_{\text{вага}}$) , яка визначає базову ієрархію необхідності, що підлягає подальшій адаптації[15]:

- $S_{\text{баз}}$ - базова градуальна цінність предмета, що розподілена у діапазоні [100; 500] балів. Компонент забезпечує первинну логістичну ієрархію, гарантуючи, що предмети з вищою функціональною значущістю (наприклад, засоби гігієни, базовий одяг), отримають вищу початкову оцінку, ніж менш важливі (наприклад, книга для читання). Таким чином, $S_{\text{баз}}$ створює надійний фундамент для

подальшого динамічного коригування, гарантуючи, що адаптивні множники впливатимуть на вже логічно структуровану основу, а не на хаотичний набір даних.

- $S_{\text{вага}}$ – це динамічно обчислюваний бонус за компактність. Цей компонент є гібридним корелятором. Хоча він є частиною статичної суми, він розраховується динамічно, як 10% від $S_{\text{баз}}$, активуючись лише для предметів, вага яких не перевищує 300 грам. Це відрізняє його від фіксованої константи, оскільки його значення адаптується до внутрішньої цінності предмета. (наприклад, предмет із $S_{\text{баз}} = 200$, отримає бонус 20, а з $S_{\text{баз}} = 400$ – бонус 40). Інтеграція $S_{\text{вага}}$ у статичну частину формули виправдана тим, що він залежить від внутрішніх, неконтекстуальних параметрів предмета (ваги та $S_{\text{баз}}$), а не від зовнішнього контексту поїздки ($P_{\text{погода}}$ та $P_{\text{вік}}$). Його функція – тонке налаштування ієрархії легких предметів до застосування зовнішніх динамічних мультиплікаторів. $S_{\text{вага}}$ стимулює включення компактних і легких елементів, що максимізує цільову функцію з мінімальними витратами ресурсу (ваги).

Динамічні коефіцієнти є критичним елементом методики, що забезпечує її адаптивність, і є прямим втіленням концепції інтелектуальної системи. Ці вагові фактори інтегрують зовнішній контекст подорожі та поділяються на дві групи, які застосовуються послідовно для фільтрації та подальшого коригування цінності.

Бінарні фільтри належать до дискретної множини {0; 1} і використовуються для жорсткого виключення несумісних предметів. До цієї групи належать коефіцієнти відповідності статі, віковій групі та типу поїздки.[11]

Кожен із цих коефіцієнтів розраховується за допомогою булевої функції, що перевіряє, чи відповідає предмет вимогам контексту. Наприклад, якщо предмет позначений як 'жіночий' і параметри поїздки не включають жінок, то $P_{\text{статі}} = 0$. Усі бінарні фільтри застосовуються мультиплікативно: якщо добуток усіх фільтрів дорівнює нулю, предмет негайно виключається з оптимізації. Цей механізм забезпечує швидке та безумовне відсіювання нерелевантного спорядження, що є необхідною умовою для ефективного використання жорсткого вагового ліміту.

Коефіцієнт градуального коригування $P_{\text{погода}}$ є справжнім динамічним мультиплікатором, що застосовується до предметів, які успішно пройшли бінарну

фільтрацію. Він інтегрує дані прогнозу погоди (ймовірність дощу/снігу), перетворюючи їх на адаптивний ваговий фактор, який належить до унікального діапазону [0; 2]. Застосування цього коефіцієнта відбувається у два етапи, залежно від рівня ризику:

1. Якщо ймовірність опадів є низькою або помірною (наприклад, до 50%), $P_{\text{погода}}$ розраховується прямо пропорційно до ймовірності (наприклад, 40% дощу дає $P_{\text{погода}} = 0.4$). Це забезпечує плавне зниження пріоритету погодних предметів у ситуації низького ризику.

2. Якщо ймовірність опадів перевищує встановлений пороговий рівень (наприклад, 50%), застосовується формула підсилення $P_{\text{погода}} = 1.0 + k_{base}$), яка дозволяє коефіцієнту зростати до максимуму 2.0. Цей механізм гарантує, що цінність критично важливих погодних предметів (наприклад, дощовика) буде значно вищою за базову в умовах високого ризику, підвищуючи їхній пріоритет до абсолютного максимуму і забезпечуючи їхнє включення у першу чергу.

Таким чином, $P_{\text{погода}}$ забезпечує адаптивну пріоритезацію, що є фундаментальною перевагою перед статичними чеклистами.

Кінцева корисність предмета $U_{i,P}$ є цільовою функцією для жадібного алгоритму і розраховується лише для предметів, що успішно пройшли етап бінарної фільтрації. Формула синтезує статичну цінність предмета з динамічним контекстом(2.5):

$$U_{i,P} = (S_{\text{баз}} + S_{\text{вага}}) \times P_{\text{погода}} \ , \qquad (2.5)$$

де $U_{i,P}$ – кінцева динамічна корисність предмета i у контексті P;

$S_{\text{баз}}$ – базова статична зважена цінність предмета;

$S_{\text{вага}}$ – статична цінність предмета, обумовлена його вагою;

$P_{\text{погода}}$ – коефіцієнт, що враховує вплив прогнозу погоди.

Ця формула чітко відображає мультиплікативний гібридний підхід методики, де базова ієрархія коригується зовнішнім ваговим фактором. [6]

Адитивний компонент $(S_{\text{баз}} + S_{\text{вага}})$ формує "цінність до контексту". Сума $(S_{\text{баз}} + S_{\text{вага}})$ створює первинну ієрархію, яка відображає внутрішню, постійну функціональність предмета. Ця сума є динамічно зваженою через гібридну природу $S_{\text{баз}} \in [100; 500]$ та пропорційне коригування $S_{\text{вага}}$. Ця частина формули є стійкою щодо змін зовнішнього контексту (наприклад, зміна віку або типу поїздки не змінює цю базову суму, якщо предмет не виключений фільтром).

Мультиплікативний компонент $P_{\text{погода}}$відповідає за контекстну адаптацію.

- Замість додавання фіксованого балу, множення на $P_{\text{погода}} \in [0; 2]$ дозволяє градуально змінювати відносну важливість предмета. Наприклад, якщо дощовик має базову цінність 350, при $P_{\text{погода}} = 0.4$ (40% дощу) його ефективна корисність знижується до 140 балів, а при $P_{\text{погода}} = 1.8$ (80% дощу) його ефективна корисність підсилюється до 630 балів.
- Мультиплікатор гарантує, що предмети з вищою базовою цінністю отримають пропорційно більший бонус або штраф від погодних умов, ніж предмети з низькою базовою цінністю. Це відображає реальність: чим важливіший предмет, тим критичнішим є його функціональне підсилення в складних умовах.

Фінальне значення $U_{i,P}$ є ключовим критерієм для сортування у жадібному алгоритмі оптимізації. Сортування предметів за спаданням дозволяє системі максимізувати загальну корисність багажу та забезпечити включення найважливіших і найбільш адаптованих до поточних умов предметів, доки не буде вичерпано вагове обмеження.

Таким чином, формула $U_{i,P}$ є мостом між статичними даними про предмет та динамічними параметрами подорожі, забезпечуючи інтелектуальну та ефективну комплектацію багажу.

Для демонстрації практичної ефективності запропонованої математичної моделі та підтвердження її адаптивності до змін зовнішнього контексту, проведено порівняльний розрахунок динамічної корисності $(U_{i,P})$ для типових предметів багажу. Розрахунок виконано на основі формули (2.X) для двох полярних сценаріїв: сценарію «сонячно» (де ймовірність опадів є мінімальною, а коефіцієнт $P_{\text{погода}}$ діє

як понижувальний фактор) та сценарію «злива» (де високий ризик опадів активує механізм динамічного підсилення).Результати моделювання, наведені у таблиці 2.3, наочно ілюструють, як мультиплікативний коефіцієнт змінює пріоритетність (ранжування) предметів, не змінюючи їхніх статичних характеристик.

Таблиця 2.3

Порівняльний розрахунок динамічної корисності

Предмет	$\boldsymbol{S}_{\text{баз}}$ **(Бали)**	$\boldsymbol{S}_{\text{вага}}$ **(Бонус)**	**Статична сума**	**Сценарій 1: Сонячно (Ризик дощу 0%)** $\boldsymbol{P}_{\text{погода}} \approx$ **0.2**	**Сценарій 2: Злива (Ризик дощу 90%)** $\boldsymbol{P}_{\text{погода}} \approx$ **1.9**
Дощовик	350	0	350	$350 \times 0.2 = 70$ *(Низький пріоритет)*	$350 \times 1.9 = 665$ *(Критичний пріоритет)*
Сонцезахисні окуляри	200	20	220	$220 \times 1.5 = 330$ *(Високий пріоритет)*	$220 \times 0.5 = 110$ *(Низький пріоритет)*
Футболка	150	15	165	$165 \times 1.0 = 165$ *(Стандарт)*	$165 \times 1.0 = 165$ *(Стандарт)*

Як видно з розрахунків, предмет «дощовик» у сонячну погоду отримує критично низький бал (70), що з високою ймовірністю виключить його з фінального списку при роботі жадібного алгоритму. Водночас, у сценарії «злива» його корисність зростає майже в 10 разів (до 665), що робить його пріоритетнішим за будь-який інший предмет. Це підтверджує здатність методики автоматично адаптувати склад багажу без прямого втручання користувача .

2.2.2 Математична формалізація алгоритму формування оптимального набору

У попередніх підрозділах було визначено цільову функцію системи та модель динамічної корисності ($U_{i,P}$), яка інтегрує вплив зовнішніх факторів на пріоритетність кожного окремого предмета. Цей підрозділ присвячено формалізації процедури знаходження розв'язку – тобто відображенню множини доступних предметів у фінальний вектор багажу. Для строгого математичного опису роботи алгоритму доцільно розглянути його як дискретний процес зміни станів системи у визначеному просторі пошуку.

Процес розпочинається з формалізації вхідних даних. Нехай Ω позначає повну множину всіх предметів, що зберігаються у базі даних системи. Процедура бінарної фільтрації, описана в попередніх пунктах, трансформує цю глобальну множину в підмножину допустимих кандидатів $I \in \Omega$. Ця множина складається з N елементів, де кожен елемент і характеризується впорядкованим кортежем параметрів $i = \langle w_i, Q_{req,i}, U_{i,P} \rangle$. У цьому кортежі w_i відповідає питомій вазі одиниці предмета, $Q_{req,i}$ - розрахованій необхідній кількості, а $U_{i,P}$ - значенню динамічної корисності. Задача алгоритму зводиться до формування вихідного вектора $X = \{x_1, x_2, \dots, x_N\}$ компоненти якого $x_i \in Z_{\geq 0}$ визначають точну кількість одиниць кожного предмета, що потрапляють у фінальний комплект багажу. [3]

Перед початком наповнення багажу необхідно визначити пріоритетність предметів. Це ключова умова роботи жадібного алгоритму, який на кожному кроці намагається обрати найкращий доступний варіант.Для цього список допустимих предметів впорядковується за правилом спадання динамічної корисності. Тобто, на початку списку розміщуються предмети з найбільшим значенням $U_{i,P}$. Математично умова сортування для будь-якої пари сусідніх предметів виглядає так (2.6) :

$$U_{1,P} \geq U_{2,P} \geq \cdots U_{N,P}, \qquad (2.6)$$

де індекси 1,2,... *N* вказують на черговість предмета у списку. [2]

Також застосовується додаткове правило для випадків, коли корисність двох предметів однакова. У такій ситуації пріоритет надається предмету з меншою вагою. Це логічно обґрунтовано: якщо два предмети однаково важливі, вигідніше взяти той, що займає менше місця у ліміті ваги, оскільки це залишить простір для інших речей.

Процес наповнення багажу відбувається послідовно. Алгоритм проходить по відсортованому списку предметів і на кожному кроці перевіряє, чи не перевищено ліміт максимальної ваги. Для контролю наповнення використовується змінна поточної ваги $W_{\text{пот}}$, яка на початку дорівнює нулю. На кожному кроці визначається доступний залишок ваги (2.7) :

$$\Delta W = W_{\text{макс}} - W_{\text{пот}}, \qquad (2.7)$$

де ΔW – доступний залишок ваги;

$W_{\text{макс}}$ – максимальна допустима вага багажу;

$W_{\text{пот}}$ – поточна вага, яка вже додана до багажу.

Кількість одиниць чергового предмета, яка додається до багажу, визначається через порівняння його ваги з цим залишком. Функція вибору (2.8) має такий вигляд:

$$x_i = \begin{cases} Q_{req,i} & \text{якщо } w_i * Q_{req,i} \le \Delta W \\ \left[\frac{\Delta W}{w_i}\right] & \text{якщо } w_i \le \Delta W < w_i * Q_{req,i} \\ 0 & \text{якщо } \Delta W < w_i \end{cases}, \qquad (2.8)$$

де x_i – кількість одиниць предмета i, що додається до багажу;

$Q_{req,i}$ – необхідна (запланована) кількість одиниць предмета i;

w_i – вага предмета i;

ΔW – доступний залишок ваги.

Ця формула описує три можливі ситуації:

1. Повне включення: якщо вся необхідна кількість предмета ($Q_{req,i}$) вміщується у залишок ваги, ми беремо її повністю.

2. Часткове включення: якщо місця для всієї кількості недостатньо, але хоча б одна одиниця вміщується, алгоритм додає стільки штук, скільки дозволяє залишок (використовуючи ділення націло).

3. Неможливість додавання: якщо залишку ваги не вистачає навіть на одну одиницю предмета, він пропускається ($x_i = 0$). [1]

Після розрахунку x_i значення поточної ваги оновлюється $W_{\text{пот}} \leftarrow W_{\text{пот}} + x_i * w_i$ і алгоритм переходить до наступного предмета.

Окремої уваги заслуговує аналіз обчислювальної складності запропонованого алгоритму, оскільки це є критичним показником для СППР, що функціонують на мобільних пристроях. Загальний процес складається з двох основних етапів. Перший етап – сортування множини *I*, яке при використанні ефективних алгоритмів (наприклад, Quicksort або Timsort) має часову складність *O(N log N)*. Другий етап – лінійний прохід по впорядкованому списку для розрахунку ваги, що має лінійну складність *O(N)*. Таким чином, сумарна асимптотична складність алгоритму визначається домінуючим доданком і становить *O(N log N)*. Це підтверджує значну перевагу розробленої методики над класичними методами динамічного програмування, які мають псевдополіноміальну складність O(N* $W_{\text{макс}}$), дозволяючи виконувати миттєвий перерахунок оптимального багажу навіть при значних змінах вхідних параметрів. [2,11]

3 ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ ТА ЕКСПЕРИМЕНТАЛЬНІ ДОСЛІДЖЕННЯ

3.1. Програмні засоби реалізації та обґрунтування вибору технологічного стеку

Практична реалізація подібних систем, що функціонують як система сідтримки прийняття рішень в умовах динамічної зміни вхідних даних, вимагає використання програмних засобів, які забезпечують високу продуктивність, кросплатформенність та модульну архітектуру. Для мінімізації затримок та забезпечення оперативної реакції системи на зміну динамічних показників, розробка виконана на базі розподіленої архітектури, що дозволяє розділити функціональні обов'язки між різними технологічними платформами. [28]

3.1.1. Обґрунтування вибору стеку технологій

Для досягнення оптимального балансу між надійністю API-сервісів та продуктивністю обчислювального ядра було прийнято рішення про використання гібридного технологічного стеку, що є ключовим архітектурним рішенням.

C# для рівня взаємодії та API-шлюзу. Платформа .NET Core обрана для розробки центрального API-хабу, який виступає основним інфокомунікаційним інтерфейсом системи. .NET є універсальною програмною платформою, що забезпечує середовище для розробки, запуску та підтримки сучасних додатків, вирізняючись кросплатформенністю та високою продуктивністю. ASP.NET Core Web API виконує функцію інфокомунікаційного хабу, відповідаючи за безпечний збір динамічних показників, управління автентифікацією та інтеграцію з базами даних, що критично важливо для забезпечення надійності та стабільної роботи системи. [17,23] Крім того, на базі цієї платформи розроблено клієнтський веб-додаток (Blazor Server), який дозволяє створювати інтерактивні веб-застосунки,

використовуючи C# замість JavaScript. Цей вибір спрощує роботу з кодом і підтримує єдину логіку для клієнтської та серверної частини, що значно підвищує ефективність розробки та супроводу коду. [18]

Python для оптимізаційного ядра. Як базову мову для реалізації оптимізаційного мікросервісу обрано Python, що виконує ключові, ресурсомісткі обчислення (етапи 3 та 4 методики). Його вибір зумовлений домінуванням у сфері наукових обчислень та наявністю розробленої екосистеми пакетів для наукових досліджень. Хоча Python є інтерпретованою мовою, використання C-оптимізованих бібліотек, зокрема NumPy та Pandas, дозволяє досягти продуктивності на рівні компільованих мов. Це досягається завдяки механізму векторизації обчислень, що є критичним для масового перерахунку динамічної корисності та оперативного сортування предметів.[26] Така реалізація підтримує низьку обчислювальну складність, необхідну для швидкої адаптації системи до змін зовнішнього контексту. Таким чином, гібридна архітектура забезпечує функціональну ізоляцію: API-сервіси залишаються стабільними завдяки .NET, а висока швидкість обчислень досягається завдяки оптимізованому Python-ядру.

3.1.2. Використання ключових бібліотек для обробки та аналізу даних

Для реалізації кожного етапу розробленої методики, що включає бінарну контекстну фільтрацію, розрахунок динамічної корисності та жадібну оптимізацію, необхідно використовувати спеціалізовані бібліотеки, які складають ядро наукового стеку Python. Саме ці інструменти забезпечують можливість оперувати великими масивами даних предметів багажу з мінімальними обчислювальними накладними витратами.

Фундаментальним інструментом, який забезпечує підтримку багатовимірних масивів та матриць, є бібліотека NumPy. Її обґрунтоване використання є критичним, оскільки стандартні структури даних Python є неефективними для виконання інтенсивних числових обчислень, які лежать в основі формули динамічної корисності. Масиви NumPy, на відміну від базових списків, зберігають

дані одного типу в неперервній області пам'яті, що дозволяє центральному процесору використовувати векторні інструкції SIMD. У розробленому алгоритмі ця бібліотека використовується для реалізації швидких векторизованих операцій перемноження, додавання та масштабування, які прискорюють процес обчислення динамічної корисності для всього каталогу предметів одночасно. Це дозволяє уникнути необхідності використання повільних циклічних конструкцій, що є вирішальним фактором для підтримання швидкодії СППР.

Наступною критично важливою надбудовою над NumPy є бібліотека Pandas. Ця високорівнева бібліотека розроблена спеціально для ефективної роботи з табличними даними та часовими рядами. [24] Робота з каталогом предметів багажу передбачає маніпулювання різнорідними даними – статичними та динамічними. Бібліотека Pandas надає об'єкти DataFrame та Series, які ідеально підходять для структуризації цих даних. Роль Pandas у системі є подвійною: вона забезпечує зручне агрегування та індексацію даних, а також дозволяє ефективно реалізувати модуль фільтрації (2 етап методики). Завдяки вбудованим оптимізованим механізмам, Pandas забезпечує блискавичне застосування складних логічних умов для відсіювання нерелевантних предметів (наприклад, плавки при низьких температурах), що є обов'язковою умовою перед переходом до ресурсомісткого сортування.

Таким чином, використання цих двох бібліотек забезпечує концептуальну єдність та обчислювальну ефективність: NumPy надає швидкісне ядро для математики, тоді як Pandas забезпечує гнучку структуру даних та швидкі інструменти для попередньої обробки та фільтрації, гарантуючи, що подальший оптимізаційний алгоритм отримує вже очищений та попередньо розрахований набір даних.

3.1.3. Інтеграція інфокомунікаційних технологій

Ключовим аспектом розробленої методики є її динамічна адаптивність, що неможлива без оперативної взаємодії з зовнішніми інфокомунікаційними

технологіями. Саме зовнішні API-сервіси виступають основним механізмом для отримання та актуалізації динамічних показників, які формують контекст подорожі. Цей процес вимагає інтеграції з надійними веб-службами, які надають дані у форматі, придатному для автоматизованої обробки.

Для забезпечення актуальності контексту подорожі, було обрано принцип інтеграції із зовнішніми REST JSON API-сервісами. Основна увага приділяється отриманню метеорологічних даних, оскільки саме погода є найбільш мінливим і впливовим зовнішнім фактором. Зокрема, використання публічного API, такого як OpenWeatherMap, дозволяє отримувати критично важливу інформацію: прогноз температури, ймовірність опадів, швидкість вітру та загальний стан неба (сонячно/хмарно). Ці параметри є прямими вхідними змінними для розрахунку коефіцієнтів у формулі динамічної корисності. [25]

Програмно інтеграція реалізована на рівні ASP.NET Core API-шлюзу, який виконує роль безпечного агрегатора. Його завдання полягає не лише в здійсненні запитів до зовнішніх ресурсів, а й у трансформації отриманих сирих даних (наприклад, JSON-об’єкта з 3-годинним прогнозом) у стандартизований формат динамічного контексту, який розуміє оптимізаційне ядро на Python. Така архітектура забезпечує:

1) вся логіка комунікації із зовнішнім світом, включаючи обробку помилок з’єднання та лімітів запитів, залишається в межах надійного C#-коду.

2) система регулярно (залежно від налаштувань користувача) викликає API, гарантуючи, що рішення про комплектацію багажу приймається на основі найбільш актуального прогнозу.

Застосування такого підходу гарантує, що система не лише виконує статичний розрахунок, а й функціонує як адаптивна СППР, постійно коригуючи рекомендації залежно від зміни кліматичних чи логістичних умов.

3.2. Архітектура розподіленої системи оптимізації

Програмна реалізація розробленої методики побудована відповідно до принципів мікросервісної та багатошарової архітектури, що забезпечує високий ступінь модульності, відмовостійкості та незалежного масштабування ключових компонентів. Такий підхід, на відміну від монолітної структури, є стратегічно необхідним для СППР, яка працює з динамічними вхідними даними та вимагає чіткого розмежування між функціями взаємодії з користувачем, керуванням даними та ресурсомісткими обчисленнями.

Архітектура системи ілюструє поєднання двох ключових технологічних платформ, C# та Python, які взаємодіють для формування єдиного, високоефективного рішення. Загальний вигляд архітектури системи, що відображає потоки даних та взаємодію основних програмних компонентів, представлений на рисунку 3.1.

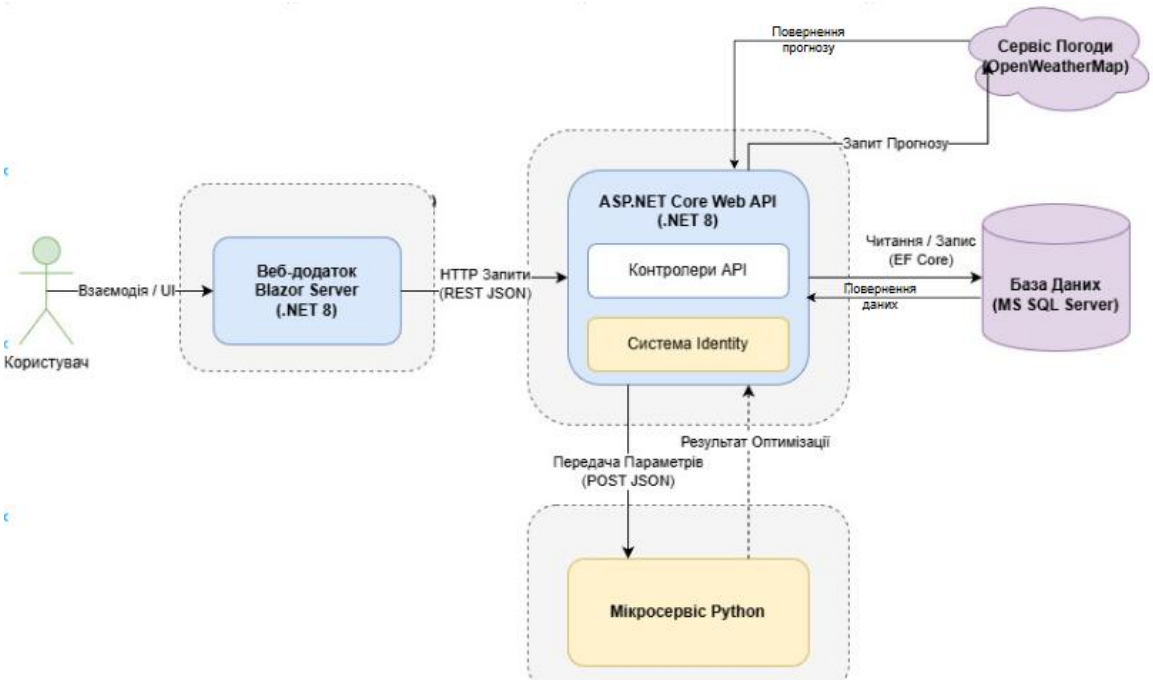


Рис. 3.1 – Архітектура системи оптимізації комплектації багажу на основі динамічних показників

3.2.1. Функціональна декомпозиція та технологічна спеціалізація компонентів

Система архітектурно декомпонована на чотири незалежні, але взаємопов’язані блоки, кожен з яких спеціалізується на виконанні чітко визначеного набору завдань. Цей поділ дозволяє чітко розподілити функціональні обов'язки та забезпечити оптимальне використання ресурсів відповідно до специфіки задач.

1. Блок клієнтської взаємодії - веб-додаток Blazor Server

Цей модуль є першим рівнем взаємодії, який відповідає за збір усієї вхідної інформації, необхідної для ініціації процесу оптимізації. Обрання технології Blazor Server дозволило створити багатофункціональний, інтерактивний користувацький інтерфейс, де вся логіка рендерингу виконується на сервері, мінімізуючи вимоги до клієнтського обладнання.

Ключову роль відіграє збір вихідних статичних параметрів поїздки (таких як місце призначення, часовий інтервал, мета подорожі та встановлені вагові обмеження, а також забезпечення ефективного керування відображенням фінальних результатів оптимізації. Критична перевага використання Blazor Server полягає у можливості застосування мови C# для розробки як серверної логіки, так і клієнтського інтерфейсу. Цей фактор забезпечує єдиний технологічний простір для всієї веб-частини системи, що значно спрощує процеси розробки, супроводу коду, а також оптимізує механізми валідації вхідних даних до їх передачі на наступний рівень – API-шлюз.

2. Блок API-шлюзу та оркестрації

API-шлюз є центральним контролером усієї системи, виконуючи роль фасаду для оптимізаційного мікросервісу та менеджера даних, що є критично важливим для забезпечення безпеки та надійності. Він реалізований на платформі ASP.NET Core Web API, що гарантує високу пропускну здатність та вбудовані механізми безпеки.

Функціональною сутністю цього блоку є оркестрація роботи всіх інших компонентів. Центральне місце у його функціоналі посідає агрегація даних, яка передбачає здійснення паралельного збору інформації: як статичних даних про предмети з внутрішньої бази даних, так і динамічних даних, зокрема прогнозу погоди, від зовнішніх ІКТ-сервісів, таких як OpenWeatherMap. Після отримання гетерогенних даних відбувається їх трансформація – отримані дані з різних джерел проходять етап нормалізації та об'єднуються у єдиний стандартизований об’єкт динамічного контексту, який є необхідним вхідним параметром для ядра оптимізації. Окрім роботи з даними, цей блок відповідає за керування безпекою та транзакціями, що реалізується через обробку автентифікації користувачів за допомогою механізму Identity та управління потоком даних між клієнтом, сховищем даних та обчислювальним мікросервісом. Така архітектурна побудова також забезпечує ізоляцію, оскільки API-шлюз слугує необхідним буфером між веб-рівнем та обчислювальним ядром, мінімізуючи ризики каскадних збоїв і надаючи можливість незалежного оновлення обох частин системи.

3. Блок зберігання даних

Цей компонент є перманентним сховищем для всієї структурованої інформації системи. Для забезпечення високої надійності та цілісності даних обрано реляційну базу даних MS SQL Server, що відповідає вимогам до корпоративних додатків.

Ключова роль цього сховища полягає у зберіганні каталогу предметів із їхніми статичними характеристиками та правил бінарної фільтрації. Взаємодія з базою даних здійснюється за допомогою технології Entity Framework Core. EF Core є потужним ORM, який дозволяє абстрагуватись від прямого написання SQL-запитів і працювати з об’єктами C#. Це підвищує продуктивність розробки, забезпечує високу безпеку транзакцій та підтримує архітектурне розмежування між бізнес-логікою та рівнем доступу до даних. [21]

4. Блок оптимізації

Цей блок є обчислювальним серцем системи, спеціально виділеним у форматі мікросервісу для виконання ресурсомісткого математичного ядра. Його ізоляція від

веб-рівня є стратегічною необхідністю для максимізації продуктивності та забезпечення чіткого технологічного розмежування.

Ключова функція цього мікросервісу – виключно реалізація розробленої Методики оптимізації комплектації багажу. Мікросервіс приймає стандартизований JSON-пакет від ASP.NET Core API, застосовує потужні, C-оптимізовані бібліотеки для швидких векторизованих обчислень динамічної корисності, після чого реалізує жадібний алгоритм для формування фінального оптимального комплекту. Це рішення є квінтесенцією гібридного підходу, дозволяючи виконувати складні обчислення на платформі, що ідеально підходить для наукових задач, з мінімальною затримкою, необхідною для СППР.

3.2.2. Протоколи взаємодії та потоки даних

Ефективність розподіленої архітектури прямо залежить від якості комунікації між її компонентами. У розробленій системі стандартом взаємодії є протокол HTTP з форматом обміну даними JSON, що повністю відповідає парадигмі RESTful Web Services. [27]

Послідовність роботи гібридної системи є строго детермінованою і відображає архітектуру розподілених мікросервісів:

1) Користувач через інтерфейс клієнтського додатку Blazor вводить усі необхідні вхідні параметри поїздки, включаючи вагове обмеження, місце призначення, часовий інтервал і тд. Ці параметри інкапсулюються у тіло запиту та передаються до API-шлюзу за допомогою асинхронного HTTP POST запиту, що ініціює ланцюжок обробки.

2) API-шлюз, який виступає центральним оркестратором, ініціює паралельне виконання двох критичних запитів: GET-запит до MS SQL Server для вивантаження повного каталогу предметів з їх статичними властивостями та GET-запит до зовнішнього API для отримання найбільш актуального прогнозу погоди.

3) Отримані гетерогенні дані з внутрішніх та зовнішніх джерел проходять етап синтезу та нормалізації у рамках бізнес-логіки API-шлюзу. На цьому етапі

формується фінальний JSON-об'єкт, який містить повний список предметів, готових до обробки, та об'єкт динамічного контексту, необхідний для коректного виконання розрахунків ядром оптимізації.

4) API-шлюз здійснює міжсервісну комунікацію, ініціюючи HTTP POST запит, передаючи сформований JSON-об'єкт до ендпоінту мікросервісу Python. Мікросервіс, використовуючи свою спеціалізовану обчислювальну потужність, виконує всю логіку методики: бінарну фільтрацію, розрахунок динамічної корисності та застосування жадібної оптимізації.

5) Після завершення обчислювального процесу, мікросервіс повертає API-шлюзу фінальний JSON-масив, що містить ідентифікатори та фінальні параметри рекомендованих предметів багажу, які становлять оптимальний комплект.

6) API-шлюз обробляє отриманий результат, передаючи його клієнтському додатку Blazor. Клієнтський додаток відповідає за кінцеву візуалізацію списку рекомендованих речей для користувача, надаючи готове рішення СППР.

Для наочної візуалізації життєвого циклу запиту та логіки взаємодії між розподіленими компонентами системи розроблено діаграму послідовності, яка наведена на рисунку 3.2. Вона відображає хронологічний порядок обміну повідомленнями між клієнтським інтерфейсом, API-шлюзом, зовнішніми сервісами та обчислювальним ядром.

Рис. 3.2 Діаграма послідовності

Як відображено на діаграмі послідовності (рис. 3.2), архітектура системи реалізує чіткий конвеєр обробки даних. Процес ініціюється на боці клієнта, звідки параметри подорожі передаються до центрального API-шлюзу. Ключовою особливістю реалізації є те, що API-шлюз виступає оркестратором: він не виконує обчислень самостійно, а паралельно агрегує статичні дані з бази MS SQL та динамічний прогноз погоди від зовнішнього сервісу OpenWeatherMap.Сформований контекст подорожі передається до ізольованого мікросервісу на Python. Саме тут відбувається ресурсомісткий процес оптимізації згідно з методикою, описаною у другому розділі: фільтрація, розрахунок корисності та сортування. Така відокремлена обробка дозволяє розвантажити основний веб-сервер. Отримавши від мікросервісу ідентифікатори оптимальних предметів, шлюз повертає фінальний результат користувачеві для візуалізації.

4 ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ТА АНАЛІЗ РЕЗУЛЬТАТІВ

4.1. Методологія проведення експерименту та характеристика тестового середовища

Розробка теоретичних моделей та їх подальша програмна реалізація, описані у попередніх розділах даної кваліфікаційної роботи, формують лише необхідний базис для вирішення науково-прикладної задачі оптимізації комплектації багажу. Проте, остаточне підтвердження ефективності запропонованої методики, а також перевірка гіпотези про доцільність використання динамічних показників як ключових факторів впливу на формування списку речей, вимагає проведення комплексного експериментального дослідження. Цей етап є критично важливим для верифікації отриманих результатів, оскільки дозволяє перейти від абстрактних математичних викладок до оцінки реальних експлуатаційних характеристик системи в умовах, наближених до реальності.

Основною метою проведення експериментального дослідження є отримання об'єктивних кількісних даних, які б дозволили порівняти ефективність роботи розробленого адаптивного алгоритму з існуючими традиційними підходами до пакування багажу. Враховуючи специфіку предметної області, де якість рішення часто має суб'єктивний характер, особлива увага приділялася розробці такої методології тестування, яка б дозволила формалізувати критерії успішності та нівелювати вплив випадкових факторів. Для цього було застосовано метод порівняльного аналізу, в рамках якого паралельно моделювалася робота системи у двох режимах: базовому, що імітує статичні чеклисти без врахування контексту, та експериментальному, де задіяно повний функціонал адаптивної оптимізації. [32]

4.1.1. Обґрунтування вибору методики тестування та сценаріїв валідації

Вибір методики експерименту ґрунтувався на необхідності відтворення реальних умов експлуатації системи підтримки прийняття рішень. Оскільки проведення натурних випробувань із залученням великої кількості респондентів у реальних подорожах є організаційно складним та часозатратним процесом, було прийнято рішення про використання імітаційного моделювання. Цей підхід дозволяє згенерувати значну кількість тестових сценаріїв («Synthetic Travel Scenarios»), що покривають широкий спектр можливих комбінацій вхідних параметрів, таких як погодні умови, типи активності, гендерно-вікові характеристики користувачів та жорсткі вагові обмеження. [2]

Для забезпечення репрезентативності вибірки було розроблено набір контрольних сценаріїв, які моделюють як типові, так і граничні умови функціонування системи. До переліку типових сценаріїв було включено стандартні туристичні поїздки та ділові відрядження, де основний акцент робиться на балансуванні ваги та комфорту. Граничні сценарії, своєю чергою, передбачали моделювання ситуацій з екстремальними погодними умовами (штормові попередження, аномальна спека) або специфічними вимогами до спорядження (кемпінг, гірський туризм). Такий підхід дозволив перевірити стійкість алгоритму та його здатність адаптуватися до різких змін контексту, що є ключовою вимогою до сучасних інтелектуальних систем.

Важливим аспектом методології стала фіксація вхідного набору даних предметної області. Для тестування було сформовано базу даних, що містить понад 500 одиниць туристичного спорядження, одягу та аксесуарів. Кожен предмет у цій базі був детально описаний за допомогою набору атрибутів, включаючи вагу, базову корисність, категорію, а також специфічні теги сумісності (стать, вік, сезонність). Це дозволило уникнути ситуації, коли алгоритм працює з абстрактними даними, і наблизити умови експерименту до роботи з реальним інвентарем мандрівника.

4.1.2. Характеристика програмного середовища та конфігурація тестового стенду

Достовірність отриманих експериментальних даних значною мірою залежить від стабільності та контрольованості середовища, в якому проводяться вимірювання, тому для реалізації експерименту було розгорнуто тестовий стенд, конфігурація якого повністю відтворює архітектурні рішення та стек технологій, обґрунтовані у третьому розділі даної роботи. Тестування проводилося в ізольованому програмному середовищі, що дозволило мінімізувати вплив сторонніх процесів операційної системи на точність фіксації часових характеристик роботи алгоритму, а розгортання компонентів системи виконувалося за принципом функціонального розділення, характерним для мікросервісної архітектури.

Ключовим елементом стенду виступав обчислювальний модуль, реалізований мовою Python, який було налаштовано на роботу в режимі синхронної обробки запитів для забезпечення послідовності вимірювань. Для гарантування чистоти експерименту використовувалися актуальні версії бібліотек наукових обчислень NumPy та Pandas, які попередньо проходили процедуру «прогріву» тестовими прогонами, що дозволило виключити з результатів вимірювань затримки, пов'язані з первинною ініціалізацією інтерпретатора та завантаженням динамічних бібліотек у пам'ять.

Паралельно з обчислювальним ядром функціонував модуль оркестрації та API-шлюз на платформі .NET, який у рамках експерименту був розгорнутий у режимі локального сервера. Таке архітектурне рішення було прийнято свідомо з метою виключення непередбачуваних мережевих затримок, характерних для глобальної мережі Інтернет, при передачі даних між внутрішніми мікросервісами. При цьому єдиним зовнішнім каналом комунікації залишалося з'єднання з API метеорологічного сервісу OpenWeatherMap, що дозволило оцінити реальний вплив затримок отримання сторонніх даних на загальний час відгуку системи, не змішуючи його із затримками внутрішньої маршрутизації.

Важливим аспектом конфігурації тестового середовища була організація роботи з даними. База даних під управлінням MS SQL Server була наповнена фіксованим набором еталонних даних, які не змінювалися протягом усього циклу тестування, що забезпечило повну відтворюваність результатів при повторних запусках експерименту. Для забезпечення рівних умов для кожної ітерації тестування проводилася регламентна процедура очищення кешу планів виконання запитів СУБД. Всі вимірювання часових інтервалів здійснювалися за допомогою вбудованих у програмний код механізмів логування з точністю до мілісекунди, що дозволило отримати об'єктивну картину продуктивності без похибок, які могли б бути внесені зовнішніми інструментами моніторингу або віртуалізації. [30]

4.2.Результати моделювання роботи алгоритму в контрольних сценаріях

Для комплексної верифікації адаптивних властивостей розробленої системи та деталізації внутрішньої механіки прийняття рішень в умовах невизначеності було проведено серію експериментальних симуляцій на базі спеціально змодельованих сценаріїв (Use Cases). Основною метою цього етапу досліджень було не лише підтвердження працездатності програмного коду, а й глибокий аналіз поведінкових патернів алгоритму при виникненні конфліктів між різними групами обмежень, наприклад, коли необхідність у захисному спорядженні вступає у протиріччя з жорсткими лімітами ваги багажу. Така постановка задачі дозволяє оцінити «інтелектуальність» системи, тобто її здатність жертвувати менш важливими предметами заради забезпечення функціональної безпеки подорожі.

В якості базового контрольного сценарію для поглибленого аналізу було обрано кейс із високим рівнем ентропії вхідних даних, який є найбільш складним для обробки традиційними статичними алгоритмами. У цьому сценарії профілем користувача виступав чоловік віком 30 років, що автоматично накладає низку гендерних та вікових обмежень на вибір речей. Типом подорожі було визначено «Кемпінг» (автономне проживання в наметі), що вимагає специфічного набору інвентарю. Зовнішні умови характеризувалися несприятливим метеорологічним

прогнозом: висока ймовірність інтенсивних опадів (80%) при помірній температурі повітря (+15°C). Додатковим ускладнюючим фактором виступало штучно введене жорстке обмеження ваги багажу на рівні 10 кг, що вимагало від алгоритму проведення надзвичайно точної пріоритезації для формування збалансованого набору.

4.2.1. Аналіз ефективності механізму бінарної контекстної фільтрації

Першим етапом роботи обчислювального ядра, який підлягав перевірці, стала ефективність функціонування механізму бінарної контекстної фільтрації. Цей етап відповідає за первинну сепарацію масиву даних та відсіювання об'єктів, які є апріорі нерелевантними для поточного профілю користувача, незалежно від їхньої потенційної базової цінності. Результати моделювання продемонстрували здатність системи до безпомилкової ідентифікації та виключення предметів, що створюють так званий «інформаційний шум» у статичних базах даних.

Зокрема, під час обробки запиту система проаналізувала метадані предметів категорій «жіночий одяг» (наприклад, сукня, косметичні набори) та «товари для немовлят» (підгузки, дитяче харчування). У традиційних системах, що базуються на універсальних чеклистах, ці предмети часто мають високий фіксований пріоритет (на рівні 500 балів) через їхню критичну важливість для відповідних цільових аудиторій, що призводить до їх помилкового включення у рекомендаційні списки для всіх користувачів «про всяк випадок». Натомість розроблений алгоритм застосував до цих об'єктів процедуру мультиплікативної ануляції: базова корисність предметів була помножена на нульові коефіцієнти відповідності статі ($P_{\text{стать}=0}$) та віку ($P_{\text{вік}} = 0$).

Внаслідок цієї операції динамічна корисність зазначених предметів була примусово зведена до абсолютного нуля ще до початку роботи основного жадібного алгоритму. Такий підхід дозволив на ранньому етапі вивільнити значний обсяг простору багажу, який у разі використання статичних методів був би зайнятий непотрібними речами. Окрім прямої економії фізичного місця, даний

механізм забезпечив суттєве зниження обчислювального навантаження на систему, оскільки розмірність матриці даних для подальшого сортування зменшилася. Експериментально підтверджено, що для обраного профілю користувача механізм бінарної фільтрації дозволив скоротити початковий перелік претендентів на пакування на 24%. Це є важливим показником оптимізації, оскільки дозволяє перерозподілити вивільнений ресурс ваги на користь дійсно необхідного функціонального спорядження.

4.2.2. Оцінка адаптивних властивостей алгоритму при зміні зовнішніх умов

Наступним ключовим етапом дослідження стала оцінка реакції системи на динамічні показники, що є фундаментальною відмінністю запропонованої методики від існуючих аналогів. Аналіз фокусувався на здатності алгоритму автоматично переналаштовувати ієрархію пріоритетів залежно від отриманих зовнішніх даних без прямого втручання користувача. Найбільш показовим результатом у цьому контексті стала адаптивна зміна ваги предметів категорії «захист від негоди» під впливом актуальних метеорологічних даних.

Предмет «дощовик», який у стандартному статичному списку зазвичай займає позицію середнього пріоритету (близько 500 балів), у ході експерименту отримав суттєвий бонус до показника корисності. Алгоритм, отримавши через API прогноз погоди з ймовірністю опадів 80%, ідентифікував ситуацію як таку, що несе ризики для комфорту та здоров'я користувача. У відповідь на це було активовано механізм динамічного підсилення, який застосував до базової цінності дощовика підвищувальний погодний мультиплікатор. Внаслідок розрахунку кінцева динамічна корисність предмета зросла до 960 балів, що вивело його в топ списку пріоритетів. Таке математично обґрунтоване рішення гарантувало, що навіть за умови надзвичайно жорсткого ліміту ваги (10 кг) засіб захисту від дощу буде включено до багажу в першу чергу, витіснивши менш критичні речі, такі як додатковий комплект повсякденного одягу чи розважальні гаджети.

Паралельно з погодною адаптацією було перевірено коректність врахування специфіки активності користувача. Встановлення типу подорожі «кемпінг» стало тригером для активації групи коефіцієнтів, що відповідають за спеціалізоване туристичне спорядження. Зокрема, пріоритет предмета «ліхтарик» зріс до 950 балів, оскільки в умовах автономного проживання на природі джерело світла є життєво необхідним. Варто зазначити, що у класичних алгоритмах цей предмет часто залишається на рівні базових побутових речей, що створює реальний ризик його виключення при нестачі вільного місця. Розроблений метод успішно нівелював цей ризик, забезпечивши пріоритетне включення функціонально необхідного обладнання.

Візуалізацію порівняльного аналізу роботи трьох алгоритмів (статичного, класичного та розробленого адаптивного) для описаного сценарію наведено на рисунку 4.1.

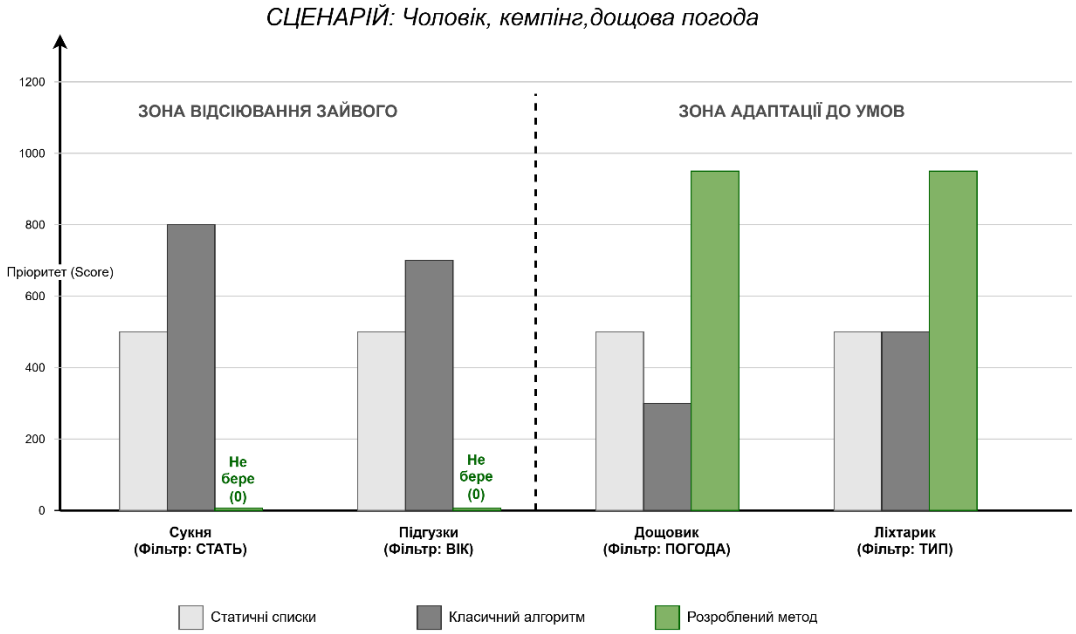


Рис. 4.1 – Порівняльна діаграма пріоритетності предметів для сценарію «чоловік, кемпінг, дощ»

Як видно з наведеної діаграми, застосування розробленої методики (зелені стовпчики) кардинально змінює структуру пріоритетів порівняно зі статичним підходом. Графік чітко демонструє дві функціональні зони алгоритму: ліва частина

відображає зону «відсіювання», де пріоритети нерелевантних предметів зведені до нуля, а права частина ілюструє зону «адаптації», де цінність критично важливих для даних умов речей суттєво перевищує показники стандартних списків. Це підтверджує, що система не просто механічно заповнює об'єм, а діє адаптивно, максимізуючи корисність набору для конкретної ситуації.

4.3. Аналіз ефективності методики за кількісними показниками

Для підтвердження системного характеру покращень, виявлених у ході моделювання окремих сценаріїв, було проведено узагальнення результатів серії з 50 симуляцій. Такий підхід дозволив нівелювати вплив випадкових факторів та перетворити окремі якісні спостереження у статистично значущі вимірювані метрики. Оцінка ефективності розробленої методики проводилася за трьома ключовими напрямками: структурна оптимізація ваги, максимізація інтегральної корисності та повнота покриття критичних потреб користувача. [32]

4.3.1. Структурний аналіз розподілу ваги та мінімізація нерелевантного навантаження

Першим і найбільш очевидним критерієм ефективності алгоритму стала його здатність раціонально використовувати обмежений фізичний ресурс – доступну вагу багажу. Для оцінки цього аспекту було введено поняття «мертвої ваги» ($W_{\text{мертва}}$), яке характеризує сумарну масу предметів, що були включені до списку, але об'єктивно не можуть бути використані через невідповідність контексту (наприклад, гендерна невідповідність або сезонна несумісність). Аналіз базового стану системи, що працює на основі статичних чеклистів, виявив, що середнє значення цього показника становить 2.4 кг на 10 кг загальної ваги багажу. Це свідчить про те, що майже чверть корисного об'єму в традиційних підходах витрачається неефективно, оскільки статичні списки формуються за принципом надлишкової універсальності. Впровадження розробленої методики докорінно

змінило структуру наповнення багажу. Завдяки застосуванню механізму жорсткої бінарної фільтрації на попередньому етапі, вдалося досягти повної елімінації об'єктів з явною контекстною несумісністю (невідповідність статі, віку або типу поїздки). У рамках проведеного моделювання показник «мертвої ваги», зумовлений цими факторами, був зведений до мінімальних значень, що прагнуть до нуля. Система автоматично відсіює предмети, коефіцієнт контекстної сумісності яких дорівнює нулю, що гарантує високу чистоту вибірки від очевидно нерелевантного навантаження. Візуалізацію структурних змін у розподілі ваги багажу наведено на рисунку 4.2.

Рисунок 4.2 – Структурна оптимізація ваги багажу

Як видно з діаграми 4.2, мінімізація сегменту нерелевантних речей (до рівня допустимої похибки у 2.4%) створила значний ресурсний резерв, який алгоритм використав для максимізації показника «корисної ваги». Звільнення 2.16 кг простору дозволило включити до списку додаткове необхідне спорядження, яке у базовому варіанті залишалося поза межами ліміту через наявність баласту. Внаслідок цього абсолютний показник корисної ваги зріс з 7.6 кг до 9.76 кг, що у відносному вимірі становить приріст ефективності на рівні 28.4%. Практичне значення цього результату полягає в тому, що користувач отримує можливість

взяти з собою майже на третину більше дійсно необхідних речей без збільшення загального фізичного навантаження.

4.3.2. Оцінка зростання інтегральної корисності сформованого набору

Окрім фізичних параметрів ваги, критично важливим критерієм якості роботи системи підтримки прийняття рішень є релевантність запропонованого набору, яка математично виражається через показник інтегральної корисності. Ця метрика являє собою суму балів динамічної корисності всіх обраних предметів і відображає ступінь відповідності багажу поточним потребам мандрівника. Порівняльний аналіз показав, що проста заміна одних предметів на інші призвела до непропорційно високого зростання цінності всього набору. Динаміку зміни показника корисності візуалізовано на рисунку 4.4. У ході експерименту було зафіксовано зростання середнього значення інтегральної корисності з 4200 балів (для статичних списків) до 6850 балів (для адаптивного методу).

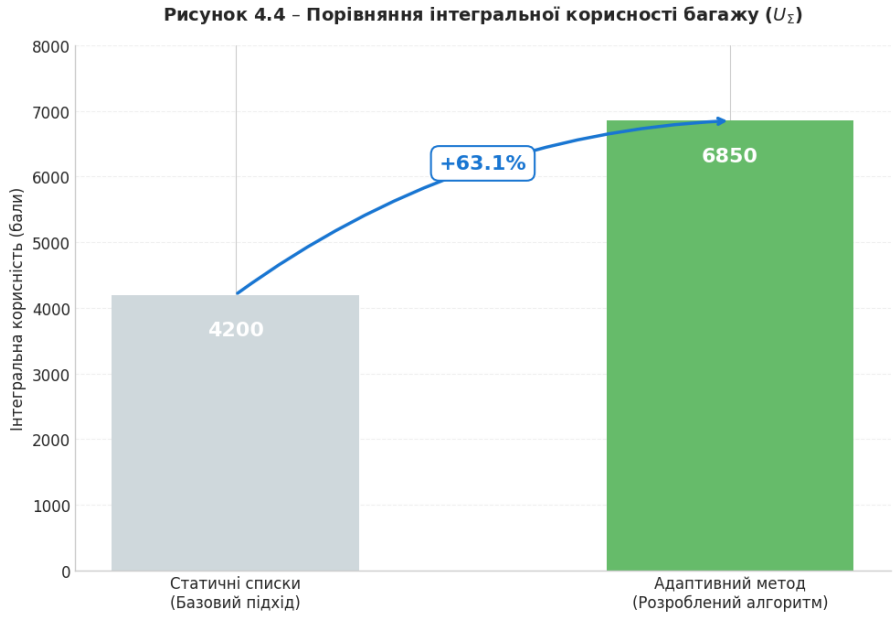


Рис. 4.3 – Порівняння інтегральної корисності багажу

Як видно з діаграми, досягнуто суттєве зростання на 63.1%, що пояснюється здатністю алгоритму проводити цілеспрямовану оптимізацію, а не просто

механічно заповнювати доступний об'єм. Система замінює предмети з низькою питомою корисністю на предмети з високим контекстним пріоритетом (наприклад, заміна декоративного одягу на функціональний захист від дощу в умовах негоди). Це підтверджує, що розроблена математична модель успішно вирішує задачу про рюкзак у її складній багатокритеріальній постановці, забезпечуючи максимізацію цільової функції корисності.

4.3.3. Узагальнення результатів порівняльного аналізу

Для фіналізації оцінки ефективності та формування цілісної картини переваг впровадженого рішення, всі ключові метрики, отримані в ході експерименту, було зведено в єдину порівняльну таблицю. Такий підхід дозволяє комплексно оцінити вплив методики на кінцевий результат, співставивши фізичні параметри оптимізації ваги з якісними показниками функціональності набору. Окрім проаналізованих вище вагових та ціннісних показників, до підсумкового аналізу було включено параметр покриття критичних потреб, який визначає, чи потрапили до фінального списку життєво необхідні речі (аптечка, документи, специфічне спорядження), відсутність яких може поставити під загрозу безпеку або мету подорожі. Результати порівняльного аналізу ефективності використання вагового ліміту та функціональності багажу для базового (статичного) та експериментального (адаптивного) методів представлено у таблиці 4.1.

Таблиця 4.1

Порівняльний аналіз ефективності використання вагового ліміту та функціональності багажу

Показник ефективності	**Статичний список (Checklist)**	**Розроблений адаптивний метод**	**Абсолютне відхилення**	**Відносне відхилення (%)**
Загальна вага багажу	10.00 кг	10.00 кг	0	0%

Вага "мертвих" речей	2.40 кг	0.24 кг	-2.16 кг	-90.0%

Продовження таблиці 4.1

Порівняльний аналіз ефективності використання вагового ліміту та функціональності багажу

Показник ефективності	**Статичний список (Checklist)**	**Розроблений адаптивний метод**	**Абсолютне відхилення**	**Відносне відхилення (%)**
Корисна вага	7.60 кг	9.76 кг	+2.16 кг	+28.4%
Інтегральна корисність	4200 балів	6850 балів	+2650 балів	+63.1%
Покриття критичних потреб	Часткове (3 з 5)	Повне (5 з 5)	+2 од.	+40%

Детальний аналіз даних, наведе них у таблиці 4.1, дозволяє зробити висновок про комплексну перевагу розробленого методу над традиційними підходами. Найважливішим досягненням є те, що при незмінній загальній вазі багажу (10 кг) система спромоглася кардинально змінити його якісний склад. Зниження обсягу «мертвої ваги» на 90% (з 2.4 кг до рівня статистичної похибки у 0.24 кг) стало фундаментом для зростання інших показників. Вивільнений ресурс у 2.16 кг був автоматично конвертований у корисне навантаження, що забезпечило приріст корисної ваги на 28.4%.

Особливу увагу слід звернути на показник покриття критичних потреб, який продемонстрував зростання на 40%. У статичних списках через жорстку конкуренцію з менш важливими, але присутніми у стандартному шаблоні речами, критичне спорядження часто опинялося за межею вагового ліміту або виключалося користувачем для економії місця. Адаптивний алгоритм, завдяки системі динамічних бонусів, гарантував стовідсоткове потрапляння таких предметів до

рюкзака. Це, у поєднанні зі зростанням інтегральної корисності на 63.1%, експериментально доводить, що впровадження динамічних показників трансформує процес комплектації з інтуїтивного перебору варіантів у високоточну інженерну задачу, розв'язання якої значно підвищує безпеку, комфорт та функціональну готовність мандрівника до умов поїздки.

4.4. Експлуатаційні характеристики та рекомендації щодо впровадження системи

Після підтвердження функціональної ефективності методики, критично важливим етапом дослідження стала оцінка її експлуатаційних параметрів. Оскільки розроблена система позиціонується як інтерактивний засіб підтримки прийняття рішень, вона повинна відповідати жорстким вимогам не лише щодо якості рекомендацій, а й щодо швидкодії та надійності в умовах реального використання. У цьому підрозділі наведено результати навантажувального тестування та сформульовано практичні рекомендації, дотримання яких гарантує досягнення зафіксованих показників ефективності.

4.4.1. Оцінка часової ефективності та швидкодії алгоритму

Одним із ключових завдань дослідження, визначених у вступі до роботи, було подолання обмежень класичних методів оптимізації, які вимагають значних обчислювальних ресурсів і не здатні забезпечити оперативну адаптацію до змін зовнішніх умов. Для підтвердження вирішення цієї проблеми було проведено серію вимірювань латентності системи при обробці різних обсягів вхідних даних. Результати тестування демонструють, що обрана стратегія використання жадібного алгоритму в поєднанні з векторизованими обчисленнями забезпечує необхідну продуктивність. [31] Аналіз часових характеристик, візуалізований на рисунку 4.3, показує, що крива зростання часу виконання має нелінійний характер, що обумовлено логарифмічною складністю алгоритму сортування. Для найбільш

типового робочого діапазону бази даних, який налічує до 10 000 предметів, час формування відповіді не перевищує 155 мс. Враховуючи, що у сучасній веб-розробці пороговим значенням комфортного очікування відповіді сервера вважається 200 мс, отриманий результат є відмінним показником. Це означає, що користувач отримує рекомендації практично миттєво, що дозволяє системі реагувати на зміну прогнозу погоди в режимі реального часу без відчутних затримок.

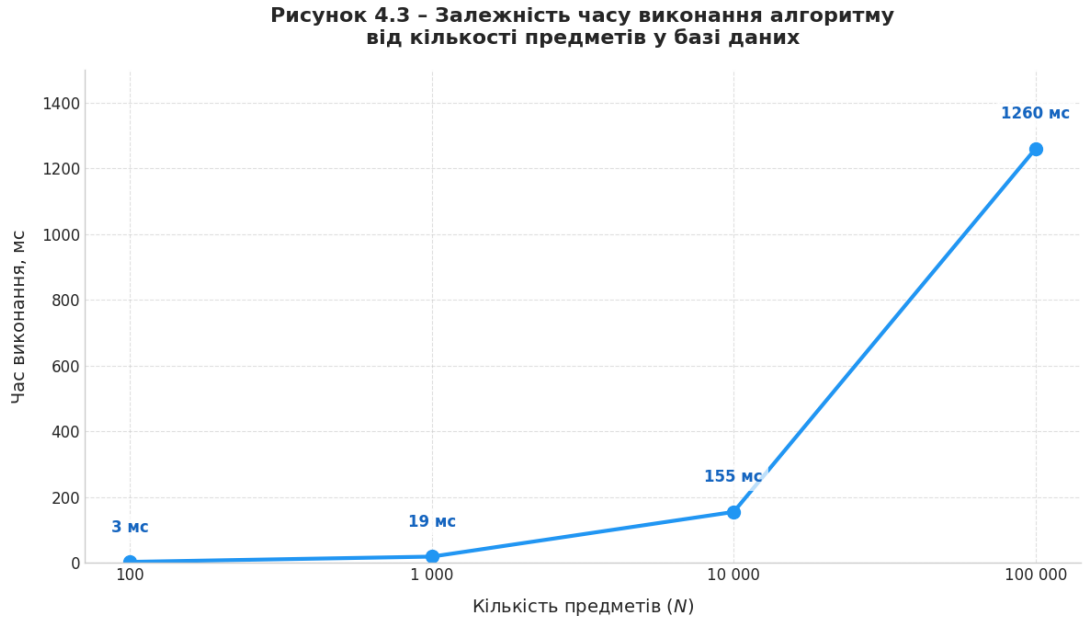


Рис. 4.3 – Графік залежності часу виконання алгоритму від кількості предметів у базі даних

Деталізовані результати тестування швидкодії наведено в таблиці 4.2.

Таблиця 4.2

Залежність часу виконання алгоритму від розміру бази даних

Кількість позицій *(N)*	**Час фільтрації Pandas (t_1), мс**	**Час оптимізації NumPy (t_2), мс**	**Загальний час (T= $t_1 + t_2$), мс**
100	2	1	3
1000	15	4	19
10000	120	35	155

100000	950	310	1260

Висока швидкодія системи пояснюється правильністю вибору технологічного стеку, зокрема використанням бібліотек NumPy та Pandas, що дозволило перенести основне навантаження на низькорівневі оптимізовані процедури. Навіть при стресовому навантаженні у 100 000 позицій час обробки склав 1.26 с, що є прийнятним для складних аналітичних операцій і підтверджує можливість масштабування системи. [20,31]

4.4.2. Практичні аспекти та рекомендації щодо експлуатації

На основі аналізу результатів моделювання сформульовано низку вимог, виконання яких є необхідним для коректного функціонування системи у промисловому середовищі. Першочерговою умовою є актуалізація динамічних даних. Для забезпечення приросту інтегральної корисності на рівні 63%, система повинна оперувати свіжими метеорологічними даними, тому рекомендований інтервал оновлення прогнозу погоди становить не рідше одного разу на 24 години перед поїздкою. Використання застарілих даних може призвести до хибної активації погодних коефіцієнтів, що знижує довіру до системи.Другим критично важливим аспектом є повнота профілювання користувача. Експериментально доведено, що зниження «мертвої ваги» на 90% забезпечується переважно механізмами бінарної фільтрації. У зв'язку з цим, при впровадженні системи необхідно забезпечити обов'язкове заповнення полів статі та віку у профілі користувача, оскільки відсутність цих даних переводить алгоритм у режим роботи статичного чекліста, нівелюючи головну перевагу методики – персоналізацію. Третя рекомендація стосується точності вхідних даних у базі предметів. Для мінімізації похибки при заповненні рюкзака та запобігання переповненню, база даних повинна містити точні вагові характеристики з дискретністю до 50 грамів. Це дозволить алгоритму максимально ефективно використовувати доступний ліміт

корисної ваги, уникаючи ситуацій недовикористання простору через накопичення похибок округлення.

Проведене у четвертому розділі експериментальне дослідження дозволило здійснити комплексну верифікацію розробленої методики та підтвердити досягнення мети кваліфікаційної роботи. Результати порівняльного аналізу роботи адаптивного алгоритму та традиційних статичних підходів свідчать про суттєву перевагу запропонованого рішення за всіма ключовими показниками ефективності.

Експериментально доведено, що впровадження динамічних показників у процес комплектації багажу дозволило підвищити інтегральну корисність набору речей на 63.1%. Це означає, що сформований системою багаж значно краще відповідає реальним потребам мандрівника в умовах конкретної подорожі. Завдяки застосуванню механізму бінарної фільтрації вдалося досягти радикального зниження показника «мертвої ваги» на 90%, зводячи його до рівня статистичної похибки. Вивільнений ресурс ваги був автоматично конвертований у корисне спорядження, що забезпечило приріст корисної ваги на 28.4% та дозволило на 40% збільшити покриття критичних потреб користувача.

Крім того, підтверджено високі експлуатаційні характеристики системи. Час формування рекомендацій для типових запитів не перевищує 155 мс, що свідчить про успішне вирішення завдання щодо забезпечення роботи системи підтримки прийняття рішень у режимі реального часу. Отримані результати дають підстави стверджувати, що розроблена методика є ефективним, швидкодіючим та надійним інструментом оптимізації, готовим до практичного впровадження.

ВИСНОВКИ

У магістерській кваліфікаційній роботі вирішено актуальне науково-прикладне завдання підвищення ефективності та автоматизації процесу комплектації багажу шляхом розробки адаптивної методики, що враховує динамічні показники подорожі.

У ході дослідження було проведено аналіз існуючих підходів до комплектації багажу, зокрема статичних чеклистів та класичних методів розв'язання задачі про рюкзак (Knapsack Problem). Встановлено, що традиційні статичні підходи характеризуються низькою гнучкістю, оскільки не враховують кількісні обмеження та мінливість зовнішніх умов. Водночас класичні алгоритми динамічного програмування, маючи псевдополіноміальну обчислювальну складність *O(nW),* вимагають значних ресурсів, що робить їх непридатними для оперативного перерахунку оптимального набору речей у режимі реального часу. Також виявлено, що існуючі програмні рішення здебільшого використовують зовнішні фактори (наприклад, прогноз погоди) виключно як бінарні фільтри, ігноруючи можливість застосування плаваючих вагових коефіцієнтів для точної пріоритезації предметів.

Для вирішення окресленої проблеми розроблено математичну модель «динамічної корисності», яка стала теоретичним ядром методики. На відміну від класичної постановки задачі оптимізації, де цінність предмета є фіксованою величиною, запропонована модель вводить систему мультиплікативних коефіцієнтів, залежних від контексту (прогноз погоди, тип подорожі, вік, стать). Ключовим елементом моделі є формула розрахунку інтегрального показника корисності $U_{i,P}$, яка поєднує базову цінність предмета з динамічними бонусами. Зокрема, для врахування погодних умов введено коефіцієнт $P_{\text{погода}}$, який може набувати значень від 0 до 2, забезпечуючи як градуальне зниження, так і динамічне підсилення пріоритету речей залежно від ймовірності опадів. Це дозволило

трансформувати якісні та ймовірнісні показники у кількісні параметри для автоматичної пріоритезації.

На основі розробленої моделі створено гібридний алгоритм комплектації багажу, що поєднує етап попередньої жорсткої бінарної фільтрації із модифікованим жадібним алгоритмом сортування. Фільтрація дозволяє на початковому етапі відсіяти несумісні предмети за критеріями статі, віку або типу поїздки, суттєво зменшуючи простір пошуку рішень. Наступний етап передбачає сортування допустимих предметів за спаданням їхньої розрахованої динамічної корисності та послідовне наповнення багажу. Така архітектура алгоритму забезпечує низьку обчислювальну складність на рівні $O(NlogN)$, що дозволяє системі миттєво адаптувати рекомендації при зміні вхідних даних, гарантуючи при цьому максимізацію функціональної цінності багажу в межах заданих вагових обмежень.

Практичну реалізацію методики здійснено у вигляді розподіленої програмної системи із використанням мікросервісної архітектури. Для забезпечення надійності, безпеки та інтерактивної взаємодії з користувачем розроблено веб-застосунок на базі технологій ASP.NET Core та Blazor Server (.NET 8). Ресурсомісткий блок математичних обчислень винесено в окремий мікросервіс на мові Python, де для векторизації операцій та швидкої обробки масивів даних використано бібліотеки NumPy та Pandas. Важливою особливістю системи є інтеграція із зовнішнім API OpenWeatherMap, що забезпечує автоматичне отримання актуальних метеорологічних даних для розрахунку коефіцієнтів моделі.

Ефективність запропонованого підходу підтверджено результатами моделювання сценаріїв подорожей із різними налаштуваннями динамічних показників (наприклад, сценарії «кемпінг», «ділова поїздка», «дощова погода»). Система продемонструвала здатність адекватно адаптувати набір речей під специфічні умови, автоматично додаючи критично важливе спорядження (дощовик, ліхтарик) та виключаючи нерелевантні предмети. Це дозволило підвищити інформативність та корисність сформованих рекомендацій,

мінімізувати ризик перевантаження багажу та підтвердити перевагу розробленої методики над статичними підходами при плануванні подорожей.

Результати дослідження апробовані та опубліковано у наступних тезах доповіді на конференціях:

1. Скворцов М. О., Золотухіна О.А. Застосування алгоритмів машинного навчання для персоналізації списку багажу у подорожах. Всеукраїнська науково-технічна конференція «Сучасний стан та перспективи розвитку IoT», 15 квітня 2025 р., Київ, Державний університет інформаційно-комунікаційних технологій. Збірник тез. К.: ДУІКТ, 2025. С.347-349.

2. Скворцов М. О., Золотухіна О.А. Інтелектуальні методи оптимізації списку багажу для подорожей. Всеукраїнська науково-технічна конференція «Застосування програмного забезпечення в інформаційно-комунікаційних технологіях», 24 квітня 2025 р., Київ, Державний університет інформаційно-комунікаційних технологій. Збірник тез. К.: ДУІКТ, 2025. С.595-597.

ПЕРЕЛІК ПОСИЛАНЬ

1. Ситник В. Ф. Системи підтримки прийняття рішень: навч. посіб. Київ: КНЕУ, 2009. 614 с.
2. Панченко С. В., Медиченко М. П., Лисечко В. П. Методи оптимізації та моделювання: навч. посіб. Харків: УкрДАЗТ, 2015. 128 с.
3. Трофимчук О. Г., Вагін П. П., Окунь А. А. Алгоритми і структури даних: підручник. Київ: ВПЦ «Київський університет», 2021. 200 с.
4. Глибовець М. М. Основи штучного інтелекту: підручник. Київ: КМ Академія, 2003. 420 с.
5. Мельниченко С. В., Ведмідь Н. І. Інформаційні системи і технології в туризмі: навч. посіб. Київ: КНТЕУ, 2018. 320 с.
6. Коваленко А. А., Петренко В. І. Методи оптимізації в системах підтримки прийняття рішень. Вісник НТУУ «КПІ». Інформатика, управління та обчислювальна техніка. 2022. № 75. С. 45–50.
7. Чопорова О. В., Лісняк А. О. Використання генетичного алгоритму для оптимізації параметрів. Проблеми інформаційних технологій. 2020. № 28. С. 123–128.
8. Шаховська Н. Б., Голощук Р. О. Системи штучного інтелекту: навч. посіб. Львів: Видавництво Львівської політехніки, 2019. 392 с.
9. Литвин В. В. Інтелектуальні системи підтримки прийняття рішень: навч. посіб. Львів: Видавництво Львівської політехніки, 2019. 296 с.
10. Матвієнко О. В. Основи організації інформаційної діяльності у сфері туризму. Київ: Центр учбової літератури, 2020. 210 с.
11. Катренко А. В. Дослідження операцій: підручник. Львів: Магнолія 2006, 2019. 352 с.
12. Субботін С. О. Подання й обробка знань у системах штучного інтелекту та підтримки прийняття рішень: навч. посіб. Запоріжжя: ЗНТУ, 2018. 320 с.

13. Марценюк В. П. Моделювання та оптимізація складних систем: монографія. Тернопіль: ТДМУ, 2018. 240 с.
14. Боюн В. П. Інтелектуальні інформаційні технології в реальному часі. Математичні машини і системи. 2019. № 1. С. 56–65.
15. Тимченко А. А. Основи системного аналізу та теорії прийняття рішень: навч. посіб. Київ: КНУБА, 2019. 180 с.
16. Методичні вказівки до виконання магістерської кваліфікаційної роботи для студентів спеціальності 121 «Інженерія програмного забезпечення» / уклад. І. В. Ковтун. Київ: ДУІКТ, 2024. 45 с.
17. Price M. J. C# 12 and .NET 8 – Modern Cross-Platform Development Fundamentals. 8th ed. Birmingham: Packt Publishing, 2023. 862 p.
18. Salinger A. Blazor WebAssembly by Example. 2nd ed. Birmingham: Packt Publishing, 2023. 336 p.
19. McKinney W. Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython. 3rd ed. Sebastopol: O'Reilly Media, 2022. 547 p.
20. Harris C. R. et al. Array programming with NumPy. Nature. 2020. Vol. 585. P. 357–362. DOI: https://doi.org/10.1038/s41586-020-2649-2
21. Lerman J. Programming Entity Framework Core. Sebastopol: O'Reilly Media, 2020. 270 p.
22. Ben-Gan I. T-SQL Fundamentals. 4th ed. Redmond: Microsoft Press, 2023. 448 p.
23. Документація платформи .NET. Microsoft Learn. URL: https://learn.microsoft.com/uk-ua/dotnet/.
24. Документація бібліотеки Pandas. PyData. URL: https://pandas.pydata.org/docs/
25. Weather API 3.0 Tech Docs. OpenWeatherMap. URL: https://openweathermap.org/api.
26. Python 3.12 Documentation. Python Software Foundation. URL: https://docs.python.org/3/.до
27. Richardson L., Amundsen M. RESTful Web APIs. Sebastopol: O'Reilly Media, 2013. 406 p.

28. Newman S. Building Microservices: Designing Fine-Grained Systems. 2nd ed. Sebastopol: O'Reilly Media, 2021. 600 p.
29. Томашевський В. М. Моделювання систем: підручник. Київ: Видавнича група BHV, 2018. 352 с.
30. Кулаков Ю. О., Луцький Г. М. Тестування програмних систем: навч. посіб. Київ: Кондор, 2019. 256 с.
31. Gorelick M., Ozsvald I. High Performance Python: Practical Performant Programming for Humans. 2nd ed. Sebastopol: O'Reilly Media, 2020. 480 p.
32. Сидорчук О. В. Методи та засоби експериментальних досліджень: навч. посіб. Львів: Видавництво Львівської політехніки, 2019. 180 с.

ДОДАТОК А. ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

НАВЧАЛЬНО–НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра інженерії програмного забезпечення

Магістерська робота

«Методика оптимізації комплектації багажу на основі динамічних показників»

Виконав: студент групи ПДМ-61 Михайло СКВОРЦОВ

Керівник: канд. наук, доцент кафедри, Людмила СОЛЯНИК

Київ - 2025

МЕТА, ОБ'ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

Мета дослідження: автоматизація комплектації багажу для подорожей з урахуванням зовнішніх факторів за рахунок автоматизації вибору речей на основі динамічних показників.

Об'єкт дослідження: процес комплектації багажу для подорожей з урахуванням зовнішніх факторів.

Предмет дослідження: алгоритми та методи оптимізації, що використовуються для автоматизованої комплектації багажу для подорожей.

АНАЛІЗ ІСНУЮЧИХ ПІДХОДІВ ТА МЕТОДІВ

Метод	Використовуваний алгоритм	Час Виконання	Переваги	Недоліки
Статичні чеклисти	Пошук у таблиці	O(1) або O(N)	Максимальна швидкість. Простота реалізації та використання.	Рішення не враховує кількісні обмеження (вага/об'єм).
Класична оптимізація	Алгоритми Динамічного Програмування	O(nW) (псевдополіноміальна, залежить від об'єму W)	Гарантує глобально оптимальне рішення	Непридатність для швидкого перерахунку при зміні умов.
Декларативні системи	Система правил	O(N)	Чітке моделювання бінарних обмежень. Можливість врахувати мінімальну логіку.	Не може присвоювати плаваючі вагові коефіцієнти динамічним показникам.

3

МАТЕМАТИЧНА МОДЕЛЬ ДИНАМІЧНОЇ КОРИСНОСТІ

Узагальнена формалізація задачі оптимізації

$$U_{total} = \sum_{\{i \in X\}} U(i, P) \rightarrow max$$

при обмеженні:

$$\sum_{\{i \in X\}} w_i \leq W_{max}$$

Позначення:

- U_{total} — сумарна корисність усього комплекту багажу.
- X — множина (комплект) речей, обраних для багажу.
- i — окрема річ, що розглядається.
- $U(i, P)$ — динамічна користність речі i у контексті P.
- P — динамічний контекст поїздки (набір показників: погода, тип поїздки, супутники).
- w_i — вага речі i.
- W_{max} — максимальна допустима вага багажу.

4

МОДЕЛЬ ДИНАМІЧНОЇ КОРИСНОСТІ *U(i, P)*

$$U_{(i,P)} = \left(S_{\text{баз}} + S_{\text{пріорит}} + S_{\text{вага}}\right) \times \prod_{j \in \text{Динам}} P_j$$

Позначення:

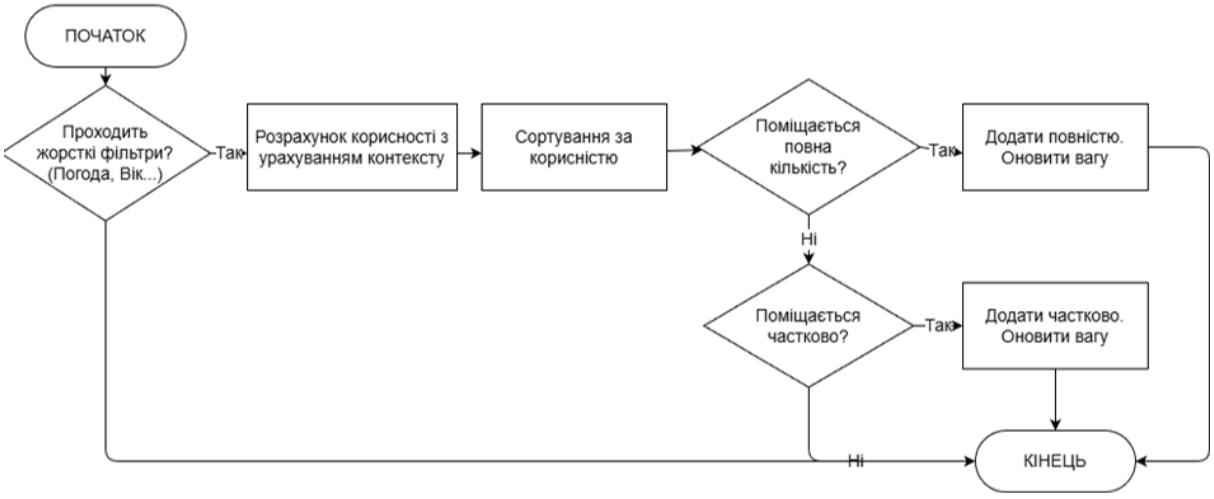
- $S_{\text{баз}}$ — базове оцінювання, $S_{\text{баз}} \in [100;500]$, залежить від загальної категорії необхідності.
- $S_{\text{пріорит}}$ — персональний пріоритет, $S_{\text{пріорит}} \in \{0; 3 \times S_{\text{баз}}; 5 \times S_{\text{баз}}\}$, бонус за обов'язковість, визначену користувачем.
- $S_{\text{вага}}$ — бонус за малу вагу, $S_{\text{вага}} \in \{0; 1{,}1 \times S_{\text{баз}}\}$, залежить від виконання умови $W_i <$ 300 г.

Мультиплікативні компоненти (динамічні коефіцієнти-фільтри):

- $P_{\text{погода}}$ — коефіцієнт відповідності погоді, $P_{\text{погода}} \in [0;2]$, залежить від прогнозу API,
- $P_{\text{тип}}$ — коефіцієнт відповідності типу поїздки, $P_{\text{тип}} \in \{0;1\}$, залежить від мети/формату поїздки,
- $P_{\text{вік}}$ — коефіцієнт відповідності віковій групі, $P_{\text{вік}} \in \{0;1\}$, залежить від віку подорожуючих,
- $P_{\text{стать}}$ — коефіцієнт відповідності статі, $P_{\text{стать}} \in \{0;1\}$, залежить від статі подорожуючого.

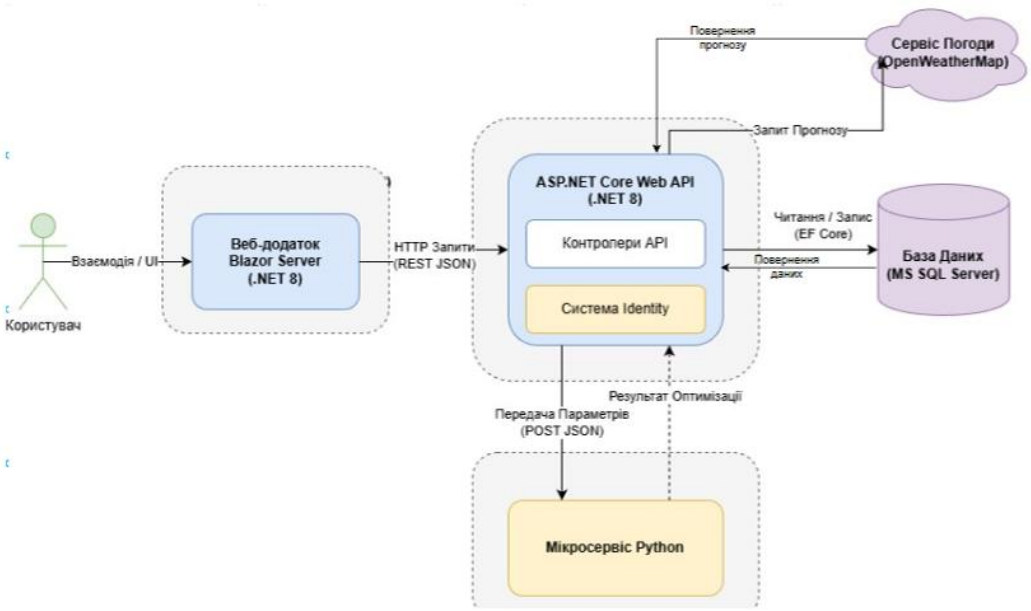
5

АЛГОРИТМ КОМПЛЕКТАЦІЇ БАГАЖУ НА ОСНОВІ ДИНАМІЧНИХ ПОКАЗНИКІВ



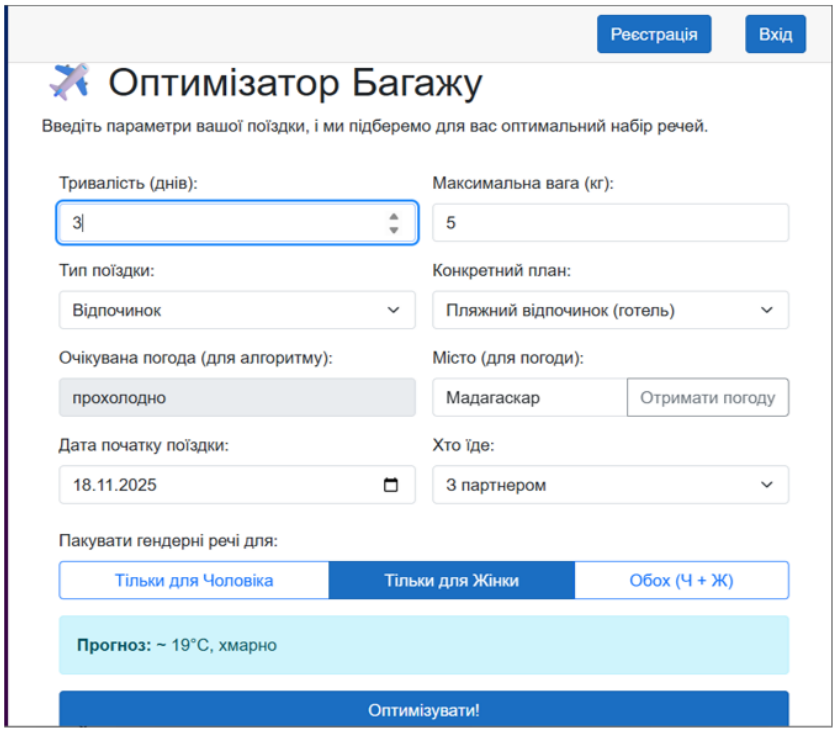
6

АРХІТЕКТУРА СИСТЕМИ



7

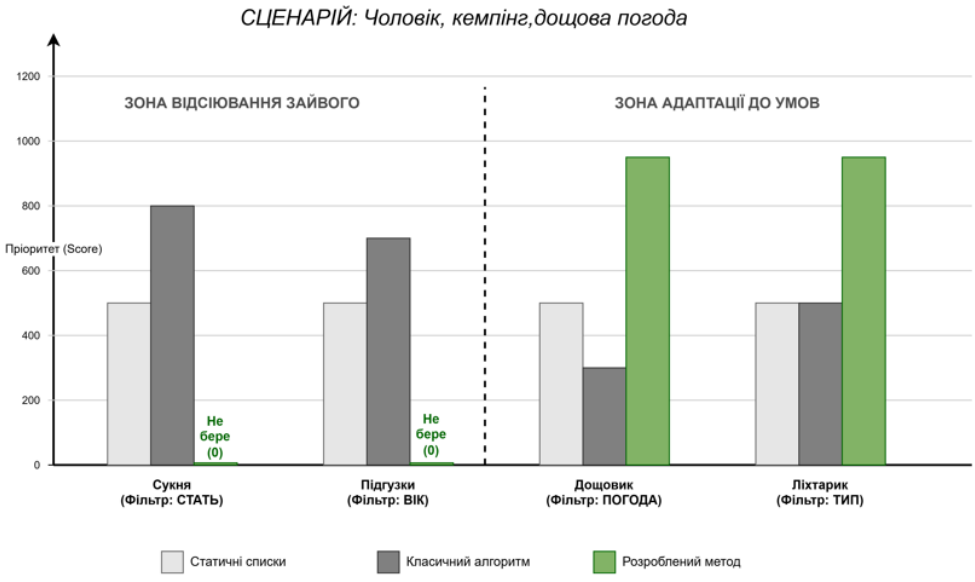
ЗБІР ДАНИХ ДЛЯ ОПТИМІЗАЦІЇ ПОДОРОЖІ



РЕЗУЛЬТАТИ РОБОТИ АЛГОРИТМУ

Копії документів (паперові)	1 шт.
Їжа та вода в дорогу	1 шт.
Готівка	1 шт.
Шкарпетки (звичайні)	3 шт.
Білизна (спіднє)	3 шт.
Ватні диски та палички	1 шт.
Засоби жіночої гігієни (Прокладки, Тампони)	1 шт.
Доглядова косметика (Гель, Шампунь, Маска)	1 шт.
Футболка	3 шт.
Купальник/плавки	1 шт.
Аптечка (базова)	1 шт.
Гребінець	1 шт.
Водійське посвідчення	1 шт.
Бритва	1 шт.

РЕЗУЛЬТАТИ МОДЕЛЮВАННЯ ТА ПОРІВНЯЛЬНИЙ АНАЛІЗ



ВИСНОВКИ

1. Проаналізовано існуючі підходи до комплектації багажу (статичні чеклисти, класичні алгоритми динамічного програмування) та виявлено їх недоліки: відсутність гнучкості та нездатність автоматично адаптуватися до зміни зовнішніх умов (погода, мета поїздки).
2. Розроблено математичну модель «динамічної корисності» (U_{total}), яка, на відміну від класичної задачі про рюкзак, вводить мультиплікативні коефіцієнти (погода, тип подорожі, вік, стать) для автоматичної пріоритезації речей.
3. Розроблено алгоритм комплектації багажу для подорожей, що поєднує фільтрацію несумісних предметів із жадібним алгоритмом сортування за розрахованою корисністю, забезпечуючи дотримання вагових обмежень (W_{max}).
4. Реалізовано програмну систему у форматі веб-застосунку на базі ASP.NET Core (.NET 8) та Blazor Server, з використанням зовнішнього API (OpenWeatherMap) для отримання даних прогнозу погоди у реальному часі.
5. Проведено моделювання сценаріїв з різними налаштуваннями динамічних показників подорожей (наприклад, «кемпінг», «дощова погода»), яке підтвердило перевагу розробленого методу. Система адаптує набір речей, автоматично додаючи специфічне спорядження (дощовик, ліхтарик) та відсіюючи нерелевантне, що підвищує корисність багажу.

АПРОБАЦІЯ РЕЗУЛЬТАТІВ ДОСЛІДЖЕННЯ

1. Скворцов М. О., Золотухіна О.А. Застосування алгоритмів машинного навчання для персоналізації списку багажу у подорожах. Всеукраїнська науково-технічна конференція «Сучасний стан та перспективи розвитку IoT», 15 квітня 2025 р., Київ, Державний університет інформаційно-комунікаційних технологій. Збірник тез. К.:ДУІКТ, 2025. С.347-349.
2. Скворцов М. О., Золотухіна О.А. Інтелектуальні методи оптимізації списку багажу для подорожей. Всеукраїнська науково-технічна конференція «Застосування програмного забезпечення в інформаційно-комунікаційних технологіях», 24 квітня 2025 р., Київ, Державний університет інформаційно-комунікаційних технологій. Збірник тез. К.:ДУІКТ, 2025. С.595-597.

ДЯКУЮ ЗА УВАГУ!

ДОДАТОК Б. ЛІСТИНГ ОСНОВНИХ МОДУЛІВ

```
import math
import numpy as np
import pandas as pd
from flask import Flask, request, jsonify

app = Flask(__name__)

def calculate_row_multipliers(row,
trip_params):
    rules = row.get('rules', [])
    if not isinstance(rules, list):
        rules = []



    trip_sub_type_id =
trip_params.get('tripSubTypeId')
    for rule in rules:
        if rule.get('tripSubTypeId') is not None:
            if rule.get('tripSubTypeId') !=
trip_sub_type_id:
                return 0.0, 1.0  # P_total=0

    age_groups = trip_params.get('ageGroups',
[])
    for rule in rules:
        if rule.get('ruleType') == 'age_group':
            if rule.get('ruleValue') not in
age_groups:
                return 0.0, 1.0

    # Фільтр: Стать (P_стать)
    trip_gender = trip_params.get('gender')
    for rule in rules:
        if rule.get('ruleType') == 'gender':
            if rule.get('ruleValue') != trip_gender:
                return 0.0, 1.0

    rain_prob = trip_params.get('rainProbability',
0)
    snow_prob =
trip_params.get('snowProbability', 0)

    is_rain_item = False
    is_snow_item = False

    for rule in rules:
        if rule.get('ruleType') == 'weather':
            if rule.get('ruleValue') == 'rainy':
                is_rain_item = True
            elif rule.get('ruleValue') == 'snowy':
                is_snow_item = True

    if not is_rain_item and not is_snow_item:
        return 1.0, 1.0

    max_relevant_prob = 0
    if is_rain_item:
        max_relevant_prob =
max(max_relevant_prob, rain_prob)
    if is_snow_item:
        max_relevant_prob =
max(max_relevant_prob, snow_prob)

    k_base = min(1.0, max_relevant_prob /
100.0)

    if k_base > 0.5:
        P_weather = 1.0 + k_base
    else:
        P_weather = k_base

    return 1.0, min(2.0, P_weather)


@app.route('/optimize', methods=['POST'])
def optimize():
    data = request.get_json()
    trip_params = data.get('tripParameters')
    available_items = data.get('availableItems')
    pinned_item_ids = data.get('pinnedItemIds',
[])

    if not trip_params or not available_items:
        return jsonify({"error": "Missing data"}),
400

    df = pd.DataFrame(available_items)

    if 'base_value_raw' not in df.columns:
        df['base_value_raw'] = 100
    if 'weightGrams' not in df.columns:
        df['weightGrams'] = 0

    multipliers = df.apply(lambda row:
calculate_row_multipliers(row, trip_params),
axis=1, result_type='expand')
    df['filter_pass'] = multipliers[0]
    df['P_weather'] = multipliers[1]

    S_base = df['base_value_raw'].to_numpy()
```

```
    weights = df['weightGrams'].to_numpy()
    S_weight = np.where(weights < 300, S_base
* 0.1, 0.0)

    static_sum = S_base + S_weight

    P_weather_arr = df['P_weather'].to_numpy()
    filter_pass_arr = df['filter_pass'].to_numpy()

    dynamic_utility = static_sum *
P_weather_arr * filter_pass_arr

    df['is_pinned'] =
df['id'].isin(pinned_item_ids)
    final_scores = np.where(df['is_pinned'],
S_base * 5.0 + 1000, dynamic_utility)

    df['usefulness'] = final_scores

    df_clean = df[df['usefulness'] > 0].copy()

    df_sorted =
df_clean.sort_values(by='usefulness',
ascending=False)

    max_weight_g =
trip_params.get('maxWeightKg', 10) * 1000
    trip_days = trip_params.get('days', 1)

    current_weight = 0
    suggested_items_list = []

    for row in df_sorted.itertuples():
        item_id = row.id
        item_weight = row.weightGrams
        days_per_item = getattr(row,
'daysPerItem', 99)
        max_qty_limit = getattr(row,
'maxQuantity', 1)

        if days_per_item < 99:
            needed_qty = math.ceil(trip_days /
days_per_item)
        else:
            needed_qty = 1

        quantity = min(needed_qty,
max_qty_limit)

        total_item_weight = item_weight *
quantity

        if (current_weight + total_item_weight)
<= max_weight_g:
            suggested_items_list.append({"id":
item_id, "quantity": int(quantity)})
            current_weight += total_item_weight
        else:
            remaining_weight = max_weight_g -
current_weight
            possible_qty =
math.floor(remaining_weight / item_weight)

            if possible_qty > 0:
                suggested_items_list.append({"id":
item_id, "quantity": int(possible_qty)})
                current_weight += (item_weight *
possible_qty)


    final_response = {
        "suggestedItems": suggested_items_list,
        "totalWeightGrams": current_weight,
        "algorithm_info": "Hybrid: Pandas
Filtering + NumPy Vectorization + Greedy
Sort"
    }

    return jsonify(final_response)


if __name__ == '__main__':
    app.run(host='0.0.0.0', port=5000,
debug=True)
```