Форма № 24

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЗАПОРІЗЬКА ПОЛІТЕХНІКА»

Факультет комп'ютерних наук та технологій
Кафедра комп'ютерних систем та мереж

Пояснювальна записка

до дипломного проєкту (роботи)
магістра
(ступінь вищої освіти)

на тему **МОБІЛЬНА СИСТЕМА ПЛАНУВАННЯ РІВНОМІРНОГО РОЗПОДІЛУ ХАРЧОВИХ ПАКУНКІВ У БАГАТОДЕННИХ ТУРИСТИЧНИХ ПОХОДАХ**

Виконав: студент 2 курсу, групи КНТ-614м
спеціальності
123 Комп'ютерна інженерія
(код і найменування спеціальності)

Освітня програма (спеціалізація)
Спеціалізовані комп'ютерні системи

БОРОВИК О. В.
(ПРІЗВИЩЕ та ініціали)
Керівник ТЯГУНОВА М.Ю.
(ПРІЗВИЩЕ та ініціали)
Рецензент РОМАНОВСЬКИЙ О.В.
(ПРІЗВИЩЕ та ініціали)

2025

Форма № 25

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет Комп'ютерних наук і технологій
Кафедра «Комп'ютерні системи та мережі»
Ступінь вищої освіти магістерський
Спеціальність 123 Комп'ютерна інженерія
(код і найменування)

Освітня програма (спеціалізація) Спеціалізовані комп'ютерні системи
(назва освітньої програми (спеціалізації)

ЗАТВЕРДЖУЮ
Зав. кафедри Кудерметов Р.К.
_____
"_15_" _жовтня___ 2025 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА

БОРОВИК Ольги Василівни
(ПРІЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Мобільна система планування рівномірного розподілу харчових пакунків у багатоденних туристичних походах
керівник проєкту (роботи) кан. техн. наук, доцент, ТЯГУНОВА Марія Юріївна
(науковий ступінь, вчене звання, ПРІЗВИЩЕ, ім'я, по батькові)
затверджені наказом вищого навчального закладу від "30" жовтня 2025 року № 491
2. Строк подання студентом проєкту (роботи) 23 грудня 2025 року
3. Вихідні дані до проєкту (роботи) опис задачі рівномірного розподілу харчових пакунків у багатоденному поході; вимоги до побудови математичної моделі; структура вхідних даних у вигляді Excel-файлів; характеристики учасників туристичної групи, включно з ваговими коефіцієнтами; необхідність розроблення алгоритму ефективного розподілу та програмної системи, що забезпечує автоматизацію цього процесу.
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1) Аналіз предметної області;
2) Реалізація мобільної системи;
3) Проведення експериментальних досліджень;
4) Результати роботи.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	ПРІЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4	ТЯГУНОВА М. Ю., доцент		
нормоконтроль	ПОЛЬСЬКА О.В., ст. викл.		

7. Дата видачі завдання 01 жовтня 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Аналіз предметної області, дослідження існуючих	05.11.2025 р.	
	методів розподілу, формування вимог та		
	обмежень		
2	Розробка математичної моделі	10.11.2025 р.	
3	Реалізація алгоритму гілок і меж	15.11.2025 р.	
4	Реалізація генетичного алгоритму	20.11.2025 р.	
5	Реалізація системи програмної системи	25.11.2025 р.	
6	Експериментальні дослідження точності,	30.11.2025 р.	
	стабільності та швидкодії алгоритмів		
7	Оформлення отриманих результатів у ПЗ	05.12.2025 р.	
8	Оформлення графічного матеріалу	10.12.2025 р.	
9	Оформлення допоміжного матеріалу	15.12.2025 р.	

Студент _____ **Ольга БОРОВИК**
(підпис) (ім'я, ПРІЗВИЩЕ)

Керівник проєкту (роботи) _____ **Марія ТЯГУНОВА**
(підпис) (ім'я, ПРІЗВИЩЕ)

РЕФЕРАТ

ПЗ: 101 с., 16 рис., 22 табл., 3 дод., 27 джерел.

РОЗПОДІЛ НАВАНТАЖЕННЯ, ЗАДАЧА ПРО РЮКЗАК, МАТЕМАТИЧНА МОДЕЛЬ, МЕТОД ГІЛОК І МЕЖ, ГЕНЕТИЧНИЙ АЛГОРИТМ, МОБІЛЬНА СИСТЕМА, ПРОЦЕСОР

Об’єкт дослідження – процес планування розподілу ваги харчових пакунків між учасниками автономного багатоденного походу.

Предмет дослідження – алгоритмічні методи забезпечення рівномірного розподілу навантаження з урахуванням заданих обмежень.

Метою роботи є автоматизація процесу рівномірного розподілу харчових пакунків у багатоденному поході шляхом створення мобільної системи FoodCalc Mobile та експериментальної оцінки алгоритмів, що забезпечують точність і стабільність цього розподілу.

У першому розділі було проаналізовано предметну область та актуальність задачі. У результаті побудовано математичну модель задачі оптимізації розподілу пакунків по рюкзаках з урахуванням обмежень.

У другому розділі описано розробку мобільної системи на базі алгоритму гілок і меж та генетичного для розв’язання задачі розподілу ресурсів.

У третьому розділі описано експериментальні дослідження, а саме: оцінювання адекватності моделі, параметри середовища, апаратні ресурси, дані для тестування та типові сценарії моделювання.

У четвертому розділі наведено аналіз отриманих результатів, зроблено висновки щодо ефективності реалізованих методів та надано візуальну демонстрацію функціонування мобільної системи.

ABSTRACT

Explanatory note to the master’s work: 101 p., 16 figures, 22 tables, 3 appendix, 27 sources.

LOAD DISTRIBUTION, KNAPSACK PROBLEM, MATHEMATICAL MODEL, BRANCH-AND-BOUND METHOD, GENETIC ALGORITHM, MOBILE SYSTEM, PROCESSOR

The object of the study is the process of planning the distribution of food packages among participants of an autonomous multi-day expedition.

The subject of the study is algorithmic methods that ensure an even distribution of load under specified constraints.

The aim of the work is to automate the process of balanced distribution of food packages in a multi-day expedition by developing the FoodCalc Mobile system and by experimentally evaluating the algorithms that provide accuracy and stability of this distribution.

The first chapter analyzes the problem domain and the relevance of the task. As a result, a mathematical model of the optimization problem for allocating packages among backpacks under given constraints was constructed.

The second chapter describes the development of the mobile system based on the branch-and-bound algorithm and a genetic algorithm for solving the resource allocation problem.

The third chapter presents the experimental studies, including the evaluation of model adequacy, environmental parameters, hardware resources, test datasets, and typical simulation scenarios.

The fourth chapter provides an analysis of the obtained results, formulates conclusions regarding the effectiveness of the implemented methods, and offers a visual demonstration of the mobile system’s functionality.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

API	–	Application Programming Interface
BnB	–	Branch and Bound
DP	–	Dynamic Programming
GA	–	Genetic Algorithm
JAR	–	Java ARchive
KP	–	Knapsack Problem
LTS	–	Long-Term Support
MKP	–	Multiple Knapsack Problem
MVP	–	Model–View–Presenter
NP	–	Nondeterministic Polynomial
SDK	–	Software Development Kit
UI	–	User Interface
XML	–	eXtensible Markup Language

ВСТУП

Сучасні тенденції розвитку туристичних, волонтерських та гуманітарних ініціатив засвідчують зростання потреби в ефективному плануванні ресурсів під час автономних багатоденних місій. За умов обмеженого доступу до зовнішнього забезпечення саме раціональний розподіл ваги спорядження між учасниками є ключовим фактором, що визначає безпеку, фізичну витривалість групи та успішність подолання маршруту. Традиційні підходи до формування індивідуальних наборів продуктів ґрунтуються на ручних оцінках, які не враховують складної комбінації чинників: різної тривалості споживання харчових пакунків, взаємозалежності добових навантажень, індивідуальних фізіологічних відмінностей учасників та необхідності забезпечення монотонного зменшення ваги рюкзаків протягом походу. Саме тому актуальним є створення мобільної системи планування рівномірного розподілу харчових пакунків у багатоденних туристичних походах, здатної забезпечити точні та відтворювані рішення без потреби в зовнішніх обчислювальних ресурсах.

У цих умовах актуальним є створення формалізованих моделей та обчислювальних методів, здатних забезпечити об’єктивний, збалансований і відтворюваний розподіл ресурсів. Додаткової важливості проблема набуває завдяки можливості застосування аналогічних методів у військовій логістиці, рятувальних операціях та автономних місіях, де точність і стабільність рішень мають критичне значення. Практична цінність такої системи полягає у здатності оперативно формувати збалансовані плани в польових умовах, без підключення до мережі, що робить її корисною як для туристичних груп, так і для підрозділів, що діють у зоні обмеженої логістики. Наукова новизна роботи полягає у формалізації постановки задачі з урахуванням багатоденної структури пакунків, фізіологічних коефіцієнтів учасників і вимоги монотонного спадання навантаження, а також у порівняльному аналізі двох різних підходів

до оптимізації для використання на мобільних пристроях. Ця робота спрямована на розв’язання зазначеної проблеми шляхом побудови математичної моделі задачі розподілу пакунків, реалізації відповідних алгоритмічних підходів та створення повноцінної програмної системи, придатної до використання на мобільних пристроях у реальних умовах.

Метою роботи є автоматизація процесу рівномірного розподілу харчових пакунків у багатоденному поході шляхом створення мобільної системи FoodCalc Mobile та експериментальної оцінки алгоритмів, що забезпечують точність і стабільність цього розподілу. Для досягнення поставленої мети необхідно виконати такі завдання:

– проаналізувати предметну область і сучасні наукові підходи до задач оптимального розподілу ресурсів;

– сформулювати постановку задачі розподілу харчових пакунків у багатоденних туристичних походах та її математичну модель;

– спроєктувати мобільну систему;

– обрати алгоритми розв’язання задачі для подальшого дослідження та реалізації;

– реалізувати обрані алгоритми у ядрі системи;

– проаналізувати отримані результати з метою вибору найбільш придатного алгоритму для інтеграції у мобільний застосунок.

Для реалізації поставлених завдань у роботі застосовуватимуться такі матеріали, методи та технічні засоби: формалізація задачі, математичне моделювання, реалізація алгоритмів гілок і меж та генетичного алгоритму, розроблення Android-застосунку і перевірка на реальних даних.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Аналіз наукових джерел і сучасних підходів

В останні роки у світі спостерігається зростання доступності подорожей і популярності активних форм туризму, зокрема пішохідних. Туристичне спорядження стає легшим і компактнішим, що дозволяє долати більші відстані та збільшувати тривалість автономних походів. Завдяки сучасним засобам комунікації туристи можуть об’єднуватися в групи, планувати маршрути та подорожувати разом незалежно від місця проживання. Зі збільшенням тривалості маршрутів та кількості учасників зростає потреба в якісній організації побуту та забезпеченні харчування, особливо в умовах автономного походу, коли група може перебувати в ізольованих від цивілізації місцевостях протягом тривалого часу, від кількох днів до кількох тижнів [1]. Кожен учасник несе на собі в рюкзаку все необхідне – їжу, медикаменти, техніку, спорядження, особисті речі. Тому важливо ефективно планувати і розподіляти вантаж, враховуючи його вагу та об’єм, щоб зменшити фізичне навантаження в екстремальних умовах.

Окрім туризму, задача ефективного планування і розподілу спорядження є актуальною також в умовах військових дій. Повномасштабне вторгнення російської федерації в Україну в 2022 році спричинило зростання потреби у засобах планування спорядження автономних місій для військових підрозділів, особливо для мобільних груп, що діють окремо від основної логістики, а також для гуманітарних місій у зонах бойових дій.

Питання продовольчого забезпечення є особливо актуальним для автономних піших походів. Індивідуальне харчування призводить до збільшення навантаження, надлишкової кількості пакування, не враховує особливості калорійності або фізичного стану учасника, його складно адаптувати до спільного приготування. Перелічені аспекти є критичними у складних умовах – на великих висотах або під час тривалих автономних

переходів, де обмежено ресурси води, газу, бензину, автоклава чи багаття [1, 2]. Колективне харчування дозволяє краще контролювати добовий раціон, підбирати його відповідно до енергетичних потреб учасників, зменшити вагу, об'єм і кількість пакунків, ефективно використовувати ресурси для приготування їжі. Водночас така централізація породжує нову складність – необхідність ефективного розподілу спільного вантажу між рюкзаками туристів [3, 4]. У багатоденних походах, де передбачено десятки пакунків і різне меню на кожен день, кількість можливих комбінацій розподілу зростає експоненційно [5]. Навіть при ретельному ручному плануванні складно збалансувати рівномірність навантаження, враховуючи об’єм пакунків, індивідуальні можливості учасників, а також щоденне монотонне зменшення ваги рюкзаків унаслідок споживання їжі. Такі практичні обмеження обґрунтовують доцільність математичної формалізації задачі. Її можна розглядати як задачу ефективного розподілу з часовими й фізичними обмеженнями, що є узагальненням задачі про множинні рюкзаки МКР [6, 7, 8]. Додатково слід урахувати умови: фіксоване закріплення пакунків за конкретним учасником і днем споживання, монотонне зменшення навантаження та обмежені фізичні можливості туристів [9, 10]. Формалізований опис цього процесу в поєднанні з можливостями програмної реалізації дозволяє автоматизувати його виконання [5, 11, 12]. Таким чином, постає потреба у побудові моделі, що враховує реальні умови розподілу продуктів для туристичного походу та формулюється в термінах комбінаторної оптимізації [3, 7]. Її програмна реалізація, зокрема у вигляді мобільного застосунку, дозволить спростити процес розподілу навантаження на етапі підготовки до походу, зробити його об’єктивним, надійним та гнучким щодо змін [5, 12, 13].

Специфіка задачі про рюкзак КР та її варіацій, зокрема задачі про множинні рюкзаки, широко висвітлюється в науковій літературі з теорії алгоритмів, обчислювальної оптимізації та операційного дослідження. У фундаментальних працях Kellerer, Pferschy і Pisinger [7], а також Martello і

Toth [8] запропоновано ключові алгоритмічні підходи: методи гілок і меж, динамічне програмування, апроксимації та евристики. Сучасні дослідження також демонструють ефективність застосування багатоцільових, евристичних та гібридних алгоритмів оптимізації для розв'язання задач МКР із урахуванням множинних обмежень і цільових функцій [12, 13, 14]. Крім того, Pisinger досліджує складність і класифікацію задач МКР, визначаючи, які з них є найскладнішими для реальних обчислень [3]. МКР активно застосовується в задачах логістики, розподілу ресурсів і планування завдань, особливо в автономних умовах.

Одним із прикладних застосувань МКР є планування спорядження для автономних військових операцій. В публікації [4] розглядається задача формування вантажу для групи морських піхотинців у місіях без логістичної підтримки. Задача формалізується як МКР із додатковими обмеженнями: передаваність предметів, групове користування, обов’язковість ресурсів тощо. Метою є проведення оцінювання рівня самозабезпеченості для конкретної місії – чи може загін морських піхотинців діяти автономно, на скільки днів вистачить ресурсів, скільки учасників потрібно, і який обсяг спорядження необхідний. У статті запропоновано модель оцінки рівня самозабезпеченості підрозділу, яка передбачає три окремі критерії: порогова самозабезпеченість – чи здатен підрозділ морських піхотинців повністю забезпечити себе необхідними ресурсами для виконання місії; ступінь самозабезпеченості – яку частку від бажаного спорядження реально взяти, враховуючи обмеження; тривалість самозабезпеченості – максимальна кількість днів автономної місії.

Утім, ця модель все ж не враховує кілька критично важливих особливостей, які є ключовими для задачі розподілу продуктів у багатоденному туристичному поході [9]:

– часову структуру;

– монотонне спадання;

– рівномірний розподіл навантаження;

– фіксованість пакунків;

– нерівномірність денного раціону;

– рівноцінність пакунків.

Часова структура означає, що кожен пакунок має добові частини споживання, які можуть відрізнятися за вагою і впливають на навантаження протягом кількох днів. Монотонне спадання передбачає, що вага рюкзака кожного туриста має зменшуватися щодня відповідно до фактичного споживання продуктів, і не може збільшуватися в жоден із наступних днів походу. Рівномірний розподіл навантаження означає, що добова вага рюкзаків повинна відповідати індивідуальним коефіцієнтам навантаження туристів, а відхилення від цільового значення не повинно перевищувати встановлений допустимий поріг. Фіксованість пакунків полягає в тому, що всі денні частини одного й того самого пакунка мають бути призначені одному туристу, оскільки один фізичний пакунок не можна поділити між різними учасниками, а також кожен учасник несе лише свою частку, без подальшого обміну. Нерівномірність денного раціону відображає те, що маса продуктів у різні дні може відрізнятися, що створює нерівномірні добові навантаження та ускладнює досягнення балансу. Рівноцінність пакунків означає що кожен пакунок має унікальну цінність і приймає участь у розподілі, він не може бути відкинутий, або взятий до розподілу двічі.

Подібні підходи можуть бути адаптовані до потреб українських сил оборони. Туристичний контекст задачі, представлений у цій роботі, в поєднанні з розглянутою моделлю задачі формування вантажу для підрозділу морських піхотинців можуть слугувати основою для розробки автоматизованого рішення для планування розподілу ресурсів для військових операцій.

В актуальній проблемі розподілу продуктів у багатоденному туристичному поході розглядається узагальнення МКР, де пакунки мають часову залежність споживання, а метою є збалансувати навантаження між учасниками походу так, щоб у кожен день навантаження спадало відповідно їхньому коефіцієнту. Додатково враховується допустиме відхилення у вазі,

проте відсутня цінність вантажу (всі пакунки мають бути враховані), а також кількість учасників і кількість днів походу не залежить від кількості вантажу.

Таким чином, задача розподілу продуктів у багатоденному туристичному поході є розширенням задачі самозабезпечення з акцентом на часову динаміку, справедливий розподіл, фіксованість пакунків, кількості учасників і днів, що не охоплено в попередніх дослідженнях. Це дозволяє точніше моделювати реальні умови багатоденного автономного походу.

1.2 Постановка задачі дослідження

Задача ефективного розподілу ваги продуктів між учасниками багатоденного туристичного походу є складною як з практичної, так і з алгоритмічної точки зору. Вона поєднує логістичні, фізіологічні та обчислювальні аспекти, що потребують комплексного підходу до формалізації та вирішення.

Нехай маємо n учасників походу, кожен із яких має індивідуальний коефіцієнт навантаження (наприклад, 0.8, 1.0, 1.2), що враховує фізичну підготовку, стан здоров’я тощо. Похід триває D днів. Для кожного дня заздалегідь сплановане меню, на основі якого формується список пакунків. Кожен пакунок має наступні характеристики: назва, вага, день використання або діапазон днів (якщо використовується кілька разів протягом декількох днів), об’єм (опціонально).

Основна складність полягає у великій кількості факторів, які необхідно врахувати для забезпечення справедливого й зручного навантаження для кожного туриста [9]. Насамперед, початкове навантаження кожного учасника має відповідати його коефіцієнту, тобто розподіл повинен бути приблизно пропорційним, щоб врахувати різні фізичні можливості і рівень підготовки учасників. При розподілі потрібно також враховувати фізичний об’єм пакунків,

щоб уникнути ситуацій, коли хтось несе легкі, але надто об’ємні продукти. Щоб врахувати об’єм можна застосувати коефіцієнт об’ємної ваги до ваги пакунку до початку розподілу пакунків по рюкзаках. Протягом кожного дня вага продуктів у рюкзаках повинна зменшуватися монотонно, відповідно до споживання. Добове навантаження кожного учасника кожного дня має залишатись у межах допустимого відхилення (наприклад, ±10%) від цільового навантаження. Після початкового розподілу пакунки закріплюються за учасниками до кінця походу і не передаються між учасниками один одному. Деякі пакунки можуть використовуватись протягом кількох днів. Наприклад, упаковка тушенки вагою 500 гр може розподілятись по 100 гр на день протягом 5 днів, або по 110, 110, 100, 90, 90 гр відповідно. У такому разі пакунок має бути закріплений за одним учасником, і його залишкова вага має враховуватись у розподілі на відповідні дні. Враховуючи фізичні можливості середньостатистичної людини у гарній спортивній формі та умови середньої складності рельєфу, можна встановити практичне обмеження на тривалість походу. Зокрема, припускаємо, що кількість днів D, протягом яких учасник походу несе всі необхідні продовольчі пакунки, не перевищує 15. Таким чином, у формальній постановці задачі додається обмеження $D \leq 15$, яке відображає реалістичність моделі. Оскільки задача має прикладне спрямування, час пошуку рішення має бути прийнятним для практичного використання, наприклад, у мобільному застосунку.

Кількість можливих комбінацій розподілу експоненційно зростає зі збільшенням кількості учасників, днів і пакунків. Вручну знайти оптимальний або навіть прийнятний варіант розподілу дуже складно. У зв'язку з цим виникає необхідність формалізувати задачу розподілу у вигляді задачі комбінаторної оптимізації, зокрема, задачі про рюкзак, або про множинні рюкзаки. Ці задачі належать до класу NP-повних, тобто не мають відомих поліноміальних алгоритмів вирішення, що суттєво ускладнює пошук ефективного рішення за прийнятний час [5, 11]. Така складність зумовлює доцільність побудови наближених або евристичних моделей для практичного застосування.

1.3 Формалізація задачі та побудова математичної моделі

Опишемо математично наявну проблему. У нашій задачі задамо наступні позначення:

n – кількість пакунків,

m – кількість туристів, кожен турист несе свій один рюкзак;

D – кількість днів походу, цілочисельне значення $1 \leq D \leq 15$;

g_i – фактична маса пакунка i , i $(\overline{1,n})$;

β_i – коефіцієнт об'ємної ваги, що враховує незручність транспортування об'ємних, але легких пакунків;

w_i – ефективна вага пакунка i, $w_i = g_i \cdot \beta_i$;

d – день походу, d $(\overline{1,D})$;

λ_j – коефіцієнт навантаження туриста j, j $(\overline{1,m})$;

W_j – цільове навантаження для туриста j на початок походу;

$x_i \in \{0,1\}$ – булева змінна, яка дорівнює 1, якщо пакунок i закріплено за туристом j.

1.3.1 Формальна модель задачі КР

Постановка задачі про рюкзак (0-1) [6]. Розглядається множина об'єктів (пакунків), кожен з яких характеризується двома параметрами: вагою w_i та деякою числовою характеристикою v_i, яка може бути інтерпретована як доцільність, пріоритет або корисність пакунка. Наявний рюкзак із максимально допустимою вантажопідйомністю W. Необхідно обрати підмножину об'єктів таким чином, щоб їхня загальна вага не перевищувала W, а сумарна характеристика v_i була максимальною.

Формальну модель задачі КР можна представити наступним чином (1.1):

$$\max \sum_{i=1}^{n} v_i \cdot x_i \qquad (1.1)$$

за умов (1.2):

$$\sum_{i=1}^{n} w_i \cdot x_i \leq W, x_i \in \{0,1\}, \forall i \in \{1,\dots,n\} \quad (1.2)$$

де $x_i = 1$, якщо предмет *i* включено до рюкзака;

$x_i = 0$, якщо предмет *i* не включено до рюкзака.

Проаналізуємо проблему на відповідність класичній задачі КР.

У таблиці 1.1 показано відповідність наявної задачі для туристів класичній задачі про рюкзак.

Таблиця 1.1 – Відповідність наявної задачі класичній задачі КР

Елемент	Knapsack Problem	Наявна задача	Відповідність
Обмеження по вазі	Рюкзак вантажопідйомністю W	Вантажопідйомність туриста W_j	Наявна
Кількість рюкзаків	1	M	Відсутня
Мета	Максимізувати корисність	Забезпечити рівномірний розподіл з урахуванням коефіцієнтів	Відсутня
Вага предметів	Враховуємо	Враховуємо	Наявна
Пакунок включено чи ні	$x_i \in \{0,1\}$	Пакунок закріплено за одним з рюкзаків $x_i \in \{0,1\}$	Часткова
Максимізація цінності	$\max \sum_{i=1}^{n} v_i \cdot x_i$	Немає явної цінності – потрібна рівність розподілу	Відсутня
Множина предметів	Маємо	Маємо	Наявна
Обмеження використання предмета з множини: тільки 1 раз	Маємо	Маємо	Наявна
Часовий аспект	Ні	Так: дні похода	Відсутня

Проміжний висновок: класична задача рюкзака частково відповідає актуальній проблемі розподілу. Проте, до класичної 0-1 КР не зводиться.

1.3.2 Формальна модель задачі МКР

Для кращої відповідності задачі розподілу харчових пакунків розглянемо задачу про множинні рюкзаки, як варіант задачі про рюкзак, у якій маємо кілька рюкзаків з окремими обмеженнями на вагу.

Формулювання МКР [7]. Маємо:

n – кількість предметів (у нашому випадку – пакунків);

m – рюкзаків;

w_i – вага i-ого предмету;

v_i – цінність i-ого предмету;

W_j – обмеження по вазі j-го рюкзака.

Необхідно призначити кожен предмет не більше ніж одному рюкзаку та максимізувати загальну цінність усіх предметів, розподілених між рюкзаками.

Формальна модель МКР [7]:

$$\max \sum_{j=1}^{m} \sum_{i=1}^{n} v_i \cdot x_{ij}, \tag{1.3}$$

за умов (1.4) та (1.5):

$$\sum_{i=1}^{n} w_i \cdot x_{ij} \leq W_j, \ \forall j \in \{1, \dots, m\}; \tag{1.4}$$

$$\sum_{j=1}^{m} x_{ij} \leq 1, \forall i \in \{1, \dots, n\}, \ x_{ij} \in \{0, 1\}, \tag{1.5}$$

де $x_{ij} = 1$, якщо предмет i включено до рюкзака j;

$x_{ij} = 0$, якщо предмет i не включено до рюкзака j.

У таблиці 1.2 показано відповідність наявної задачі для туристів задачі про рюкзак.

Маємо очевидні відмінності [15]. У задачі розподілу продуктових пакунків між рюкзаками туристів відсутній параметр цінності v_i, тобто всі пакунки однаково необхідні.

Таблиця 1.2 – Відповідність наявної задачі класичній задачі МКР

Елемент	МКР	Наявна задача	Відповідність
Обмеження по вазі	Рюкзак вантажопідйомністю W_j	Вантажопідйомність туриста W_j	Наявна
Кількість рюкзаків	m	M	Наявна
Мета	Максимізувати цінність усіх предметів, розподілених між рюкзаками	Забезпечити рівномірний розподіл з урахуванням коефіцієнтів і днів використання	Відсутня
Вага предметів	w_i	w_i	Наявна
Пакунок включено чи ні	$x_{ij} \in \{0,1\}$	$x_{ij} \in \{0,1\}$	Наявна
Цінність предмета	v_i	Немає	Відсутня
Множина предметів	n	N	Наявна
Обмеження використання предмета з множини: тільки 1 раз	Маємо	Маємо	Наявна
Часовий аспект	Ні	Так: дні похода	Відсутня

В актуальній задачі відсутня функція вигоди. Метою стає мінімізація сумарного дисбалансу навантаження між учасниками. Тобто потрібно розподілити пакунки так, щоб кожен учасник ніс свою справедливу частину – пропорційно до своїх можливостей. В нашій задачі кожен пакунок має прив’язку до дня використання. Пакунки мають бути розподілені так, щоб кожного дня навантаження кожного туриста поступово спадало за рахунок споживання.

Проміжний висновок: таким чином, задача розподілу пакунків для туристичного походу є узагальненням МКР з додатковими обмеженнями: часова залежність пакунків, монотонне спадання ваги рюкзака, балансування ваги замість максимізації вартості.

1.3.3 Побудова математичної моделі задачі дослідження

Розрахуємо цільові значення для задачі розподілу пакунків. Позначимо:

L – сумарна вага всіх пакунків продуктів для походу;

$\mathrm{L_d}$ – сумарна вага продуктів, що мають бути спожиті у день d;

w_{id} – частина ваги пакунка i, яка споживається в день d; при $\mathrm{w_{id}} = 0$, у день d пакунок не використовується;

W_{jd} – цільове навантаження для туриста j на початок дня d.

Для подальшого формального опису задачі розглянемо ключові залежності між пакунками, днями споживання та туристами [15].

Загальна вага всіх пакунків для всіх днів:

$$L = \sum_{i=1}^{n} w_i = \sum_{d=1}^{D} \sum_{i=1}^{n} w_{id} \text{ при } w_i = \sum_{d=1}^{D} w_{id}. \quad (1.6)$$

Загальна вага продуктів, що будуть спожиті у день d:

$$L_d = \sum_{i=1}^{n} w_{id}. \quad (1.7)$$

Цільове навантаження для туриста j на початок походу визначаємо середнім навантаженням на одного туриста з урахуванням індивідуального коефіцієнта за (1.8):

$$W_j = \lambda_j \cdot \frac{L}{m}. \quad (1.8)$$

Цільове навантаження для туриста j на день d визначається (1.9):

$$W_{jd} = \lambda_j \cdot \frac{L_d}{m}, \quad (1.9)$$

де $\frac{L_d}{m}$ – це середнє навантаження на одного туриста на початок дня d, яке коригуємо відповідно до індивідуального коефіцієнта.

Опишемо часову залежність пакунків.

Пакунок можна призначити лише одному туристу, який нестиме його усі дні до повного споживання.

Значення змінної $x_{ij} = 1$ вказує на те, що пакунок присутній в рюкзаку туриста j на всіх $d \leq d_{max}(i)$.

Визначимо останній день споживання пакунку:

$$d_{max}(i) = max\ \{d|w_{id}\ > 0\}, \tag{1.10}$$

де $d_{max}(i)$ – останній день споживання пакунку i, допомагає визначити, до якого дня пакунок враховується у вантажі туриста;

$\{d|w_{id}\ > 0\}$ – множина всіх днів, у які з пакунка i щось використовується;

$max\ \{d|w_{id}\ > 0\}$ – найбільший номер дня серед тих, коли пакунок використовується.

Тобто, якщо $x_{ij} = 1$, то w_{id} входить до розрахунку вантажу туриста для кожного дня, поки від пакунка лишається ненульова вага.

Визначення днів використання пакунків потрібно, щоб правильно обчислювати вагу рюкзака в дні походу. Якщо пакунок використовується в кількох днях, то вся його вага (відповідно до залишку) має бути врахована у навантаженні до і включно останнього дня його використання.

Для забезпечення рівномірного використання пакунків і монотонного спадання ваги, можна застосувати підхід розбиття задачі на підзадачі для кожного дня, що дозволяє контролювати баланс ваги і зберігати спадання.

Опишемо такий підхід. Позначимо:

W_j' – розрахункова початкова вага туриста j;

W_{jd}' – розрахункова вага туриста j на початку дня d;

L_{jd} – сумарна вага, які туристи споживають з рюкзака туриста j в день d.

Розрахуємо початкову вагу для рюкзака туриста j (загальна вага всіх пакунків, що закріплені за туристом j на початок першого дня походу):

$$W_j' = W_{j,1} = \sum_{i=1}^{n} w_i \cdot x_{ij}. \tag{1.11}$$

Вага рюкзака туриста на початок кожного наступного дня походу визначається за (1.12):

$$W'_{jd} = W_{j,d-1} - L_{j,d-1}, \forall\, d \in \{2, \dots, D\}, \tag{1.12}$$

де навантаження туриста j в день d:

$$L_{jd} = \sum_{i=1}^{n} w_{id} \cdot x_{ij}. \tag{1.13}$$

Формула (1.12) дозволяє перевірити спадання на кожному кроці і гарантовано розподілити кожен день повністю, зберігати справедливу частку туриста відносно його загального навантаження та працювати з нерівномірною вагою по днях.

У реальних умовах походу ідеальний баланс навантаження між усіма учасниками не завжди досяжний. З практичного боку, невелике відхилення в межах 10% є майже непомітним для туриста. Наприклад, якщо споряджений рюкзак важить 18 кг, з них продуктів 5 кг, то 10% відхилення від 5 кг – це 500 гр. Перевантаження рюкзака вагою 18 кг на 0,5 кг майже не впливає на комфорт, але значно спрощує пошук прийнятного розподілу пакунків [1]. Зважаючи на комбінаторну складність задачі (кількість комбінацій зростає експоненційно з кількістю пакунків і учасників), допустима похибка дозволяє знайти рішення швидше, знизити вимоги до обчислювальних ресурсів і при цьому зберегти справедливість розподілу [11].

Введемо параметр допустимого відхилення $\sigma = 0.1$ – це похибка в 10%.

Скоригуємо обмеження на денне навантаження для туриста j з урахуванням відхилення:

$$(1 - \sigma) \cdot W_{jd} \leq W'_{jd} \leq (1 + \sigma) \cdot W_{jd}. \tag{1.14}$$

Скоригуємо обмеження на сумарне навантаження для туриста j з урахуванням відхилення:

$$(1-\sigma)\cdot W_j \le W_j' \le (1+\sigma)\cdot W_j. \quad (1.15)$$

1.3.4 Результат побудови математичної моделі задачі дослідження

Можна зазначити, що задача розподілу ваги пакунків між туристами в багатоденному автономному поході формалізується як узагальнена задача МКР з додатковими обмеженнями: часовою залежністю, монотонним спаданням ваги рюкзака та балансуванням з урахуванням фізичних можливостей учасників. Формальна модель задачі розподілу пакунків у поході можна представити наступним чином [15]. Позначення:

n – кількість пакунків;

m – кількість туристів, кожен турист несе 1 свій рюкзак;

D – кількість днів походу, цілочисельне значення $1 \le D \le 15$;

λ_j – коефіцієнт навантаження туриста j, j $(\overline{1,m})$;

w_i – вага пакунка i, i $(\overline{1,n})$;

w_{id} – вага частини пакунка i, що споживається у день d, d $(\overline{1,D})$;

$x_{ij} \in \{0,1\}$ – булева змінна, яка дорівнює 1, якщо пакунок i закріплено за туристом j, 0 – якщо пакунок i не закріплено за туристом j;

L_d – загальна вага продуктів, яку має спожити група в день d;

L_{jd} – вага продуктів, які закріплені за туристом j що будуть спожиті групою у день d;

L – сумарна вага всіх пакунків;

W_j – цільове загальне навантаження туриста j на початок походу;

W_{jd} – цільове навантаження туриста j на день d;

W_j' – розрахункове загальне навантаження туриста j на початок походу;

W'_{jd} – розрахункове навантаження туриста j на початок дня d;

σ – допустиме відхилення від ідеального навантаження $\pm 10\%$.

Узагальнюючи формалізацію МКР та описані математичні обмеження, актуальну прикладну задачу можна подати у вигляді наступної системи умов.

Призначення кожного пакунка тільки одному з туристів:

$$\sum_{j=1}^{m} x_{ij} = 1, \ \forall\, i \ \in \{1, \dots, n\}. \tag{1.16}$$

Розрахунок сумарної ваги всіх пакунків:

$$L = \sum_{i=1}^{n} w_i. \tag{1.17}$$

Розрахунок загальної ваги продуктів, що будуть спожиті у день d:

$$L_d = \sum_{i=1}^{n} w_{id}. \tag{1.18}$$

Розрахунок ваги продуктів з рюкзака туриста j, що будуть спожиті у день d за (1.19):

$$L_{jd} = \sum_{i=1}^{n} w_{id} \cdot x_{ij}. \tag{1.19}$$

Цільове початкове навантаження для туриста j:

$$W_j = \lambda_j \cdot \frac{L}{m}. \tag{1.20}$$

Розрахункове початкове навантаження туриста j:

$$W_j' = W_{j,1} = \sum_{i=1}^{n} w_i \cdot x_{ij}. \tag{1.21}$$

Перевірка початкової ваги рюкзака туриста (на старті походу) з допустимим відхиленням:

$$(1-\sigma)\cdot W_j \le W_j' \le (1+\sigma)\cdot W_j. \quad (1.22)$$

Цільове навантаження туриста j на день d:

$$W_{jd} = \lambda_j \cdot \frac{L_d}{m}. \quad (1.23)$$

Розрахункове навантаження туриста j на початок дня d:

$$W'_{jd} = W'_{j,d-1} - L_{j,d-1}, \forall\, d \in \{2,\dots,D\}. \quad (1.24)$$

Перевірка балансу ваги кожного дня для туриста j з допустимим відхиленням:

$$(1-\sigma)\cdot W_{jd} \le W'_{jd} \le (1+\sigma)\cdot W_{jd}. \quad (1.25)$$

Перевіримо що виконується необхідна умова монотонного спадання ваги. Вага рюкзака туриста j щодня спадає на величину ваги продуктів, спожитих з нього в попередній день:

$$L_{j,d-1} = W'_{j,d-1} - W'_{jd},\ \forall\, d \in \{2,\dots,D\}. \quad (1.26)$$

Умова збереження сумарної ваги всіх пакунків:

$$\sum_{j=1}^{m} W_j = L. \quad (1.27)$$

Таким чином, у результаті проведеного аналізу та математичної формалізації задачі вдалося побудувати узагальнену модель розподілу харчових пакунків у багатоденному туристичному поході. Вона враховує вагові обмеження, індивідуальні коефіцієнти навантаження, часову динаміку споживання ресурсів та допустимі відхилення від цільового навантаження. Запропонований підхід дозволяє перейти від інтуїтивного ручного планування до системного комп'ютерного моделювання, яке забезпечує ефективніший і збалансований розподіл навантаження між учасниками.

1.4 Методи та підходи до розв'язання задачі

Задача рюкзака та її розширення, зокрема МКР, є класичними прикладами NP-складних задач комбінаторної оптимізації, які широко застосовуються в логістиці, плануванні ресурсів, розподілі вантажів і фінансовому аналізі [6].

Сучасні підходи до розв'язання задач МКР умовно поділяють на такі групи [17]:

– точні методи;

– евристичні та метаевристичні методи;

– гібридні підходи.

До точних методів розв'язання задач відносять динамічне програмування, алгоритм гілок і меж, метод лінійного програмування. До евристичних та метаевристичних методів – жадібні підходи, локальний пошук, алгоритми мурашиних колоній, роїв частинок, генетичні алгоритми тощо. Гібридні підходи, які поєднують переваги обох класів, наприклад гібридизацію гілок і меж із евристичними методами.

1.4.1 Точні алгоритми: динамічне програмування та метод гілок і меж

Динамічне програмування (DP) забезпечує оптимальні рішення для невеликих або середніх розмірів задач. Воно використовує принцип оптимальності Беллмана, але складність різко зростає зі збільшенням кількості рюкзаків і елементів, що робить метод обчислювально дорогим. Через експоненційну залежність від кількості параметрів DP є неефективним для практичних багатовимірних задач [17].

Метод динамічного програмування має ряд переваг. Він гарантує отримання оптимального розв’язку, оскільки ґрунтується на принципі оптимальності Беллмана, що забезпечує точність результатів у дискретних задачах. На відміну від стохастичних підходів, рішення, отримані таким методом, є повністю відтворюваними та стабільними. Алгоритм зручний у реалізації для одно- та двовимірних постановок, зокрема для класичної задачі рюкзака, задач найкоротших шляхів чи вирівнювання послідовностей. Крім того, динамічне програмування широко використовується у навчальному та аналітичному контекстах, оскільки дозволяє дослідити внутрішню структуру задачі та характер її підзадач, а також може бути основою для побудови гібридних та евристичних методів.

Метод динамічного програмування має наступні недоліки. Йому властива висока обчислювальна складність, яка зазвичай зростає пропорційно добутку кількості елементів та ємності, що швидко робить розв’язання ресурсоємним. Значний обсяг пам’яті, необхідний для зберігання таблиць динамічного програмування, є критичним обмеженням при збільшенні розмірності задачі. Масштабованість також залишається проблемною: для багатовимірних або множинних варіантів (наприклад, з кількома обмеженнями чи великою кількістю «рюкзаків») метод практично непридатний. Додатковим ускладненням є чутливість до дискретизації параметрів, що вимагає використання грубих наближень у випадку неперервних величин. У сукупності

ці фактори роблять динамічне програмування недостатньо ефективним для задач, які потрібно розв’язувати у режимі реального часу.

Метод гілок і меж (Branch and Bound, BnB) є більш універсальним підходом, що застосовується для точного розв’язання МКР та його варіантів [17]. Його основна ідея полягає у систематичному переборі варіантів із відсіканням неперспективних гілок, які не можуть дати кращого рішення. Ефективність методу залежить від правильної побудови функцій обмеження (меж), що оцінюють потенціал кожного вузла дерева рішень.

Метод гілок і меж має низку важливих переваг, що пояснюють його широке використання в задачах комбінаторної оптимізації. Він забезпечує гарантовану оптимальність результату, оскільки при повному проходженні дерева пошуку завжди знаходить ефективне розв’язання. Підхід дозволяє контролювати точність за рахунок можливості достроково припиняти пошук, коли досягнуто прийнятної похибки від оптимуму. Метод також вирізняється гнучкістю, адже може бути адаптований до різноманітних обмежень і модифікацій, зокрема до множинних і багатовимірних варіантів задачі рюкзака. Значною перевагою є ефективне відсікання гілок завдяки використанню верхніх і нижніх меж, що суттєво зменшує розмір простору пошуку порівняно з повним перебором. Крім того, BnB добре поєднується з іншими методами, такими як лінійні релаксації, евристики чи генетичні алгоритми, що робить його придатним для побудови гібридних рішень.

Разом із тим метод гілок і меж має і суттєві недоліки. Йому властива експоненційна складність у найгіршому випадку, через що час розв’язання різко зростає зі збільшенням кількості змінних. Метод вимагає значного обсягу пам’яті, зокрема при опрацюванні великої кількості вузлів дерева пошуку, що негативно позначається на масштабованості та ускладнює застосування у задачах великої розмірності або тих, що потребують роботи у реальному часі. Якість роботи алгоритму суттєво залежить від точності оцінок меж: слабкі або неточні оцінки призводять до необхідності опрацьовувати більшу кількість гілок, що значно збільшує час обчислень. Реалізація методу також потребує

ретельного налаштування, зокрема вибору стратегії відсікання, порядку обходу дерева та формування коректних меж.

Незважаючи на ці обмеження, BnB залишається базовим методом для оцінки ефективності інших алгоритмів і є відправною точкою для порівнянь у багатьох наукових роботах [5, 6].

1.4.2 Евристичні та метаевристичні підходи

Для складних практичних задач, де рішення потребує надто багато часу, застосовуються евристичні й метаевристичні підходи, які дозволяють знайти ефективне рішення за прийнятний час [17].

Жадібний алгоритм (Greedy Algorithm) є одним із найпростіших та найшвидших підходів до побудови розв’язань у задачах оптимізації [17]. Його принцип ґрунтується на покроковому формуванні результату, коли на кожному кроці обирається найкращий доступний елемент за певним локальним критерієм, наприклад максимальним відношенням «цінність/вага».

Такий метод має дуже низькі вимоги до обчислювальних ресурсів, оскільки оперує лише поточним набором альтернатив, не потребує аналізу попередніх виборів і не переглядає вже ухвалені рішення. Для задач, у яких локально оптимальний вибір дійсно приводить до глобального оптимуму, жадібний підхід працює особливо ефективно. Також він часто застосовується як допоміжна евристика або стартове наближення у складніших методах, зокрема у гілках і межах, генетичних алгоритмах та динамічному програмуванні.

Водночас жадібний алгоритм має низку обмежень, які необхідно враховувати при його застосуванні. Найсуттєвішим недоліком є те, що локально оптимальний вибір не завжди веде до глобального оптимуму, тому якість отриманих рішень може бути далекою від найкращої. Алгоритм погано працює в задачах із вираженими взаємозалежностями між змінними, де кожне проміжне рішення впливає на подальші кроки, а також у багатокритеріальних постановках, де оптимальність не зводиться до одного показника. Оскільки метод не допускає перегляду раніше ухвалених рішень, він не здатний

коригувати помилки на попередніх етапах, через що здебільшого використовується як швидка евристична оцінка або як «наближений старт» для більш потужних алгоритмів оптимізації.

Генетичний алгоритм (Genetic Algorithm, GA) належить до метаевристичних методів оптимізації та ґрунтується на механізмах природної еволюції – доборі, схрещуванні та мутації [18]. Він добре зарекомендував себе в задачах великої розмірності, особливо тоді, коли простір можливих рішень має складну структуру, містить багато обмежень або декілька цільових критеріїв [14]. Стохастична природа підходу дозволяє алгоритму одночасно досліджувати велику кількість кандидатних рішень, що підвищує ймовірність уникнення локальних мінімумів. GA не потребує аналітичної форми цільової функції та працює лише з її числовими значеннями, завдяки чому легко адаптується до різних варіантів постановки й може бути модифікований під потреби конкретної задачі. Паралельний характер обробки популяції надає методу високу гнучкість та придатність до пошукових просторів, де класичні алгоритми виявляються неефективними.

Разом із тим, генетичні алгоритми мають низку важливих обмежень. Оскільки процес пошуку є стохастичним, результати різних запусків можуть відрізнятися, що ускладнює гарантування стабільності або відтворюваності розв’язань. Метод не забезпечує строгого досягнення глобального оптимуму і може зупинятися на субоптимальних рішеннях, особливо якщо параметри – розмір популяції, імовірності мутації та схрещування – підібрані невдало. Для задач, у яких обчислення функції пристосованості є ресурсомістким, GA може працювати повільно через необхідність багаторазового оцінювання великої кількості кандидатів.

У практичних сценаріях оптимізації цей метод часто комбінують із швидкими жадібними евристиками, які забезпечують початкове наближення, після чого генетична схема виконує глобальний пошук у складному просторі рішень.

1.5 Критерії вибору алгоритмів і обґрунтування вибору алгоритмів

З урахуванням поставленої мети – збалансований розподіл харчових пакунків між рюкзаками туристів у багатоденному поході – критерії вибору алгоритмів наведено в таблиці 1.3.

Таблиця 1.3 – Критерії вибору алгоритмів для дослідження

Критерій	Обґрунтування
Якість розподілу	Відхилення від цільової ваги має бути мінімальним для кожного дня (≤10%).
Швидкодія	Алгоритм повинен працювати в реальному часі (секунди).
Масштабованість	Можливість зміни кількості рюкзаків/днів/пакунків без втрати стабільності.
Інтеграція в систему	Легка інтеграція в мобільний застосунок на Java/Spring Boot із підтримкою експорту результатів
Відтворюваність результатів	Для наукового експерименту потрібна стабільність при повторних запусках.

Порівняння проведених у сучасних роботах вказує, що метод BnB залишається одним із найефективніших точних методів для задач МКР середнього розміру. Він використовується у практичних дослідженнях розподілу ресурсів, логістики, військових постачань та навіть автономного планування маршрутів [4].

З іншого боку, генетичний алгоритм забезпечує кращу масштабованість і стійкість при збільшенні розмірності задачі. Його ефективність у розв’язанні задач КР доведено в оглядах 2020–2025 років [14].

Таким чином, для подальшого комп’ютерного дослідження та порівняння були обрані два алгоритми: метод гілок і меж, як базовий точний підхід, та генетичний алгоритм, як представник метаевристичних методів, що забезпечує компроміс між швидкістю й якістю результатів.

Такий вибір дозволяє дослідити співвідношення між точністю й обчислювальною ефективністю для задач типу МКР.

1.6 Висновки до розділу

У першому розділі проведено аналіз предметної області та сформульовано задачу ефективного розподілу харчових пакунків у багатоденному автономному поході. Розглянуто особливості задачі: наявність багатоденних пакунків, різні фізичні можливості учасників та потребу рівномірного збалансування навантаження на кожному етапі маршруту. На цій основі побудовано математичну модель, у якій розподіл інтерпретується як узагальнення задачі МКР і враховано індивідуальні коефіцієнти навантаження туристів, обчислено цільові ваги рюкзаків та визначено допустимі відхилення, та умову монотонного зменшення ваги внаслідок споживання ресурсів.

Сформована модель відображає ключові обмеження реальної задачі, забезпечує логічну узгодженість та дає змогу застосовувати методи комбінаторної оптимізації й евристичні підходи. Узагальнення структури пакунків, днів і навантажень створює основу для подальшої реалізації алгоритмів. Отримані формалізації формують підґрунтя для розробки програмної системи, а наступний розділ присвячено проєктуванню її архітектури, структури даних і алгоритмів, що реалізують модель у вигляді мобільної системи.

2 РЕАЛІЗАЦІЯ МОБІЛЬНОЇ СИСТЕМИ

2.1 Проєктування системи

Мобільна система FoodCalc Mobile проєктується як модульна програмна платформа, що забезпечує автоматизоване формування збалансованого плану розподілу харчових пакунків між учасниками багатоденного туристичного походу [19]. З архітектурної точки зору система складається з двох основних підсистем: мобільного клієнта, який відповідає за керування сценарієм роботи

та взаємодію з користувачем, і обчислювального ядра, що реалізує математичну модель задачі та алгоритмічні процедури побудови розподілу. Подібне логічне розділення компонентів відповідає сучасним принципам проєктування програмних систем, спрощуючи тестування та супровід коду [20].

2.1.1 Структура системи

Структура програмної системи FoodCalc Mobile побудована з урахуванням принципів модульності та розділення відповідальності, що відповідає сучасним підходам до проєктування прикладного програмного забезпечення [19]. Архітектура передбачає чітке відокремлення бізнес-логіки, механізмів опрацювання даних та користувацького інтерфейсу. Таке розділення забезпечує гнучкість, розширюваність та повторне використання програмних компонентів [20].

Система складається з двох незалежних, але логічно пов’язаних підпроєктів:

– foodcalc-mobile (Java Core);

– foodcalc-android (Android MVP).

Схематично структуру системи FoodCalc Mobile наведено на рисунку 2.1.

Підпроєкт foodcalc-mobile (Java Core) реалізує доменну модель, математичну логіку розподілу пакунків та сервіси імпорту й експорту даних. Ядро постачається у вигляді JAR-бібліотеки та не містить залежностей від платформних інструментів, що робить його переносимим та придатним для використання в різних програмних середовищах.

Підпроєкт foodcalc-android (Android MVP) є мобільним клієнтом, розробленим на базі Android SDK. Він забезпечує взаємодію користувача із системою та використовує функціональність ядра шляхом інтеграції JAR-модуля. На рівні клієнта реалізовано інтерфейс користувача та базові сценарії роботи з даними, тоді як обчислювальні та алгоритмічні операції виконує саме ядро.

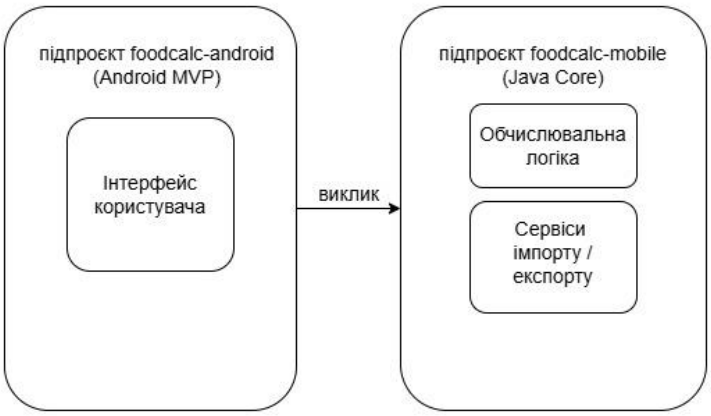


Рисунок 2.1 – Структура програмної системи FoodCalc Mobile

Таке архітектурне рішення забезпечує незалежність мобільного інтерфейсу від алгоритмічних та обчислювальних механізмів, спрощує тестування, полегшує супровід і дає можливість багаторазового використання ядра в інших середовищах (наприклад, у настільних застосунках) [20].

Далі наведено структуру доменної моделі ядра та принципи взаємодії його основних компонентів.

2.1.2 Функціонування системи

Узагальнену послідовність опрацювання даних та взаємодії між підсистемами FoodCalc Mobile подано у вигляді блок-схеми на рисунку 2.2, що відображає логіку функціонування застосунку – від моменту завантаження вхідного файлу до формування підсумкового розподілу або повідомлення про необхідність коригування даних.

Робота системи розпочинається з ініціювання мобільним застосунком процедури завантаження вхідного Excel-файлу, що містить інформацію про туристів, багатоденні пакунки та їхню структуру споживання. Після отримання файлу система переходить до перевірки коректності його структури, зокрема узгодженості кількості днів, наявності всіх обов’язкових полів та валідності представлення вагових характеристик. Якщо структура вхідного файлу не відповідає встановленим вимогам, користувач отримує повідомлення про необхідність коригування вхідних даних.

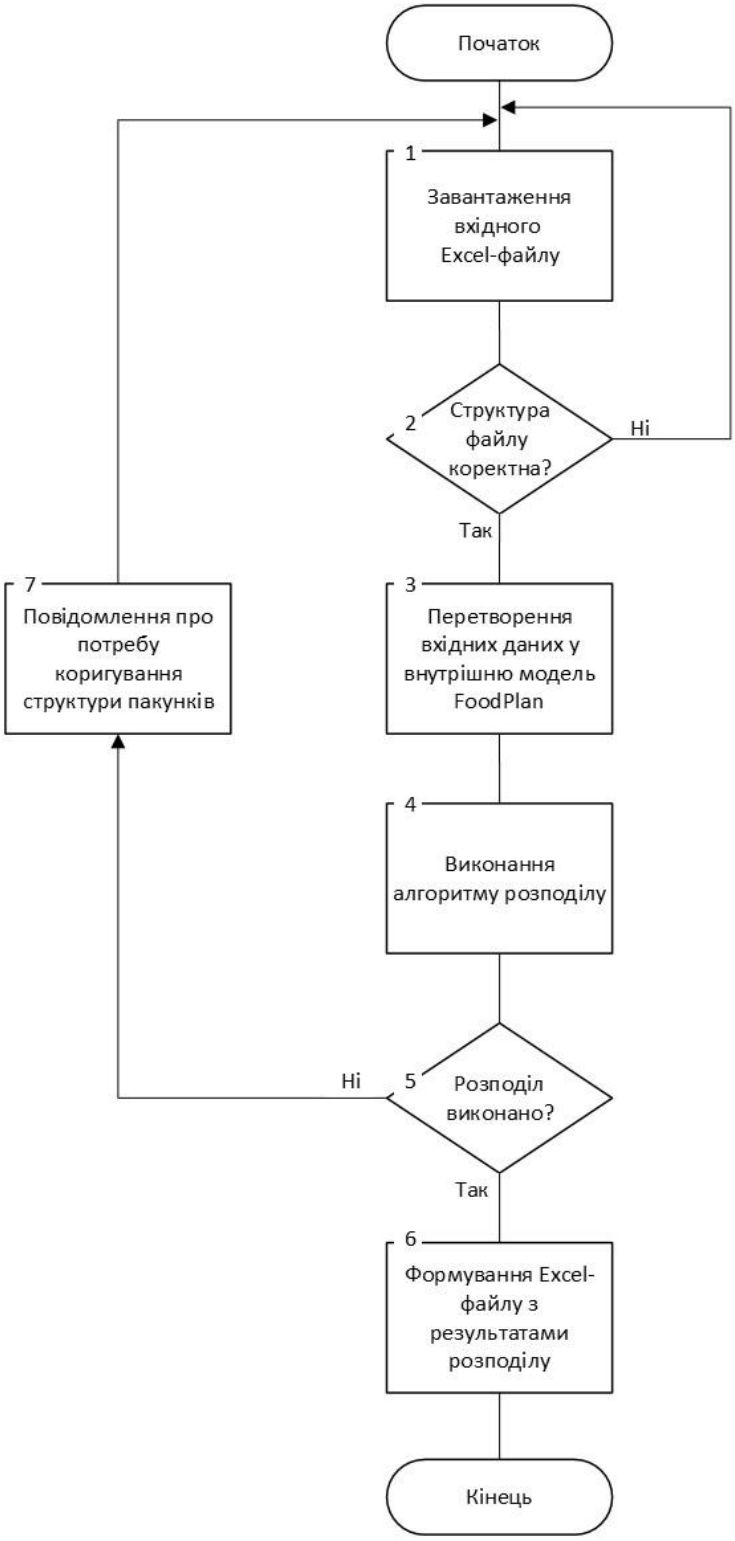


Рисунок 2.2 – Блок-схема роботи мобільної системи FoodCalc Mobile

У разі успішного проходження перевірки дані Excel-файлу трансформуються у внутрішню модель FoodPlan, яка відображає взаємозв’язки між туристами (рюкзаками), днями маршруту та пакунками, а також формує параметри, необхідні для запуску алгоритму. Після формування моделі система

переходить до етапу побудови розподілу, що виконується обчислювальним ядром за допомогою одного з двох реалізованих алгоритмів – методу гілок і меж або генетичного алгоритму [21]. Після завершення обчислень система виконує перевірку успішності формування розподілу. Якщо алгоритм не може сконструювати рішення, що задовольняє вимогу добової рівномірності та монотонного зменшення навантаження, користувачеві повертається повідомлення про потребу коригування структури пакунків.

Якщо ж алгоритм формує припустиме рішення, система переходить до завершального етапу – генерації вихідного Excel-файлу з результатами розподілу, який містить призначення пакунків між рюкзаками туристів, добові навантаження, значення відхилень та деталізовані таблиці розрахунків. Після збереження результатів у файловій системі робота застосунку завершується.

2.2 Реалізація системи

2.2.1 Доменна модель та ключові сутності системи

Доменна модель модуля FoodCalc Mobile реалізована у вигляді впорядкованої системи пакетів, кожен з яких виконує чітко визначену роль у межах загальної моделі розподілу пакунків у багатоденному автономному поході. Основу становлять пакети domain.model та domain.service, що забезпечують опис предметної області та реалізацію бізнес-логіки відповідно [21].

Пакет domain.model містить базові моделі, які відображають ключові елементи задачі:

– FoodPlan;

– Hiker;

– PlanDay;

– PackageWithProducts.

FoodPlan – інтегрована модель плану харчування, яка включає учасників, список днів та перелік пакунків.

Hiker – сутність туриста з індивідуальними параметрами, зокрема коефіцієнтом навантаження.

PlanDay – структура, що описує конкретний день походу.

PackageWithProducts – модель пакунка, представленого у вигляді відображення «день – вага», що дає змогу враховувати багатоденне використання одного пакунка.

Пакет domain.model.plan.pack містить алгоритмічні структури, що використовуються безпосередньо під час обчислень. Центральним елементом цього пакета є клас HikerState, який відображає динамічний стан туриста в процесі розподілу, включаючи його навантаження за днями, індивідуальні цільові показники та перелік призначених пакунків. Крім того, у пакеті зосереджено низку допоміжних структур, призначених для клонування станів, контролю обмежень, ведення обліку ваги та підтримки рекурсивного перебору під час виконання алгоритмів.

Побудова моделі орієнтована на специфіку задачі ефективного розподілу пакунків у багатоденному поході. Оскільки кожен пакунок може використовуватися протягом кількох днів, модель передбачає зберігання його ваги у розрізі дат. Індивідуальні коефіцієнти навантаження учасників впливають на формування персональних таргетів, що визначають очікуване денне навантаження кожного туриста. Структура моделі підтримує формування як проміжних, так і фінальних призначень, що є необхідним для реалізації рекурсивних алгоритмів, зокрема методу гілок і меж. Усі елементи моделі залишаються платформно незалежними та не містять Android- чи вебспецифічних компонентів, що забезпечує можливість повторного використання ядра в різних програмних середовищах.

2.2.2 Архітектура сервісів та логіка їх взаємодії

Пакет domain.service містить сервіси, що реалізують основні етапи оброблення даних у системі: імпорт, оптимізацію та експорт результатів. Кожен

сервіс виконує автономну функцію, а взаємодія між ними здійснюється відповідно до єдиної послідовності.

Основні сервіси ядра:

– ExcelFoodPlanParser;

– ManualBnBDistributionService;

– GeneticDistributionService;

– ExcelExportService.

ExcelFoodPlanParser – сервіс імпорту вихідного Excel-файлу, реконструкція моделі FoodPlan на основі даних про учасників, дні та пакунки.

ManualBnBDistributionService – реалізація алгоритму гілок і меж (Branch and Bound), що здійснює перебір можливих варіантів призначення пакунків і знаходить рішення з мінімальною похибкою від цільових значень.

GeneticDistributionService – реалізація евристичного підходу на основі генетичного алгоритму з використанням бібліотеки Jenetics.

ExcelExportService – сервіс формування вихідного Excel-файлу із підсумковим розподілом, деталізацією по днях та діагностичними показниками.

Взаємодія сервісів ядра відбувається за єдиною схемою, представленою на рисунку 2.3.

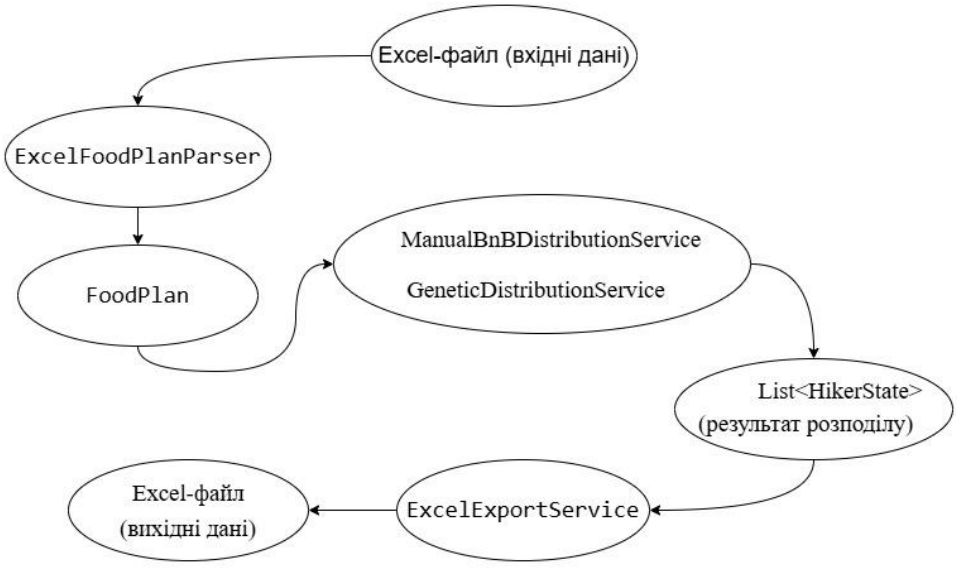


Рисунок 2.3 – Схема взаємозв’язку сервісів ядра

Мобільний застосунок FoodCalc Android реалізує платформний рівень системи й відповідає за взаємодію користувача з функціональністю ядра. Поточна реалізація представлена у вигляді мінімальної функціональної конфігурації (MVP), призначеної для перевірки базових сценаріїв роботи до впровадження повного інтерфейсу.

Основні компоненти Android-клієнта:

– MainActivity;

– інтеграція ядра (JAR-модуля);

– UI-рівень (ресурси Android);

– файлові операції Android.

MainActivity – центральна активність, що відображає UI із кнопками «Download Food Plan» і «Run Distribution»; ініціює вибір Excel-файлу за допомогою системного файлового пікера; викликає методи ядра через інтегрований JAR-модуль; обробляє результат і ініціює збереження Excel-звіту.

Бібліотека foodcalc-mobile-1.0-core.jar підключена до проєкту Android через директорію app/libs. Це дозволяє виконувати всі обчислення локально на мобільному пристрої.

Рівень користувацького інтерфейсу включає XML-макети, графічні ресурси та обробники подій, що формують взаємодію користувача із функціональністю застосунку.

Механізми оброблення файлів у мобільному застосунку реалізовано через стандартні інструменти Android API, зокрема системний файловий пікер, засоби роботи з потоками введення та механізм збереження результатів через інтегровані діалоги операційної системи.

Усі етапи обчислень відбуваються в ядрі, тоді як FoodCalc Android виконує лише:

– отримання та передавання вхідного Excel-файлу в ядро;

– запуск алгоритму через JAR-модуль;

– збереження сформованого Excel-файлу на пристрої;

– відображення повідомлень і результатів для користувача.

Послідовність виконання сценарію в мобільній системі представлена на рисунку 2.4.

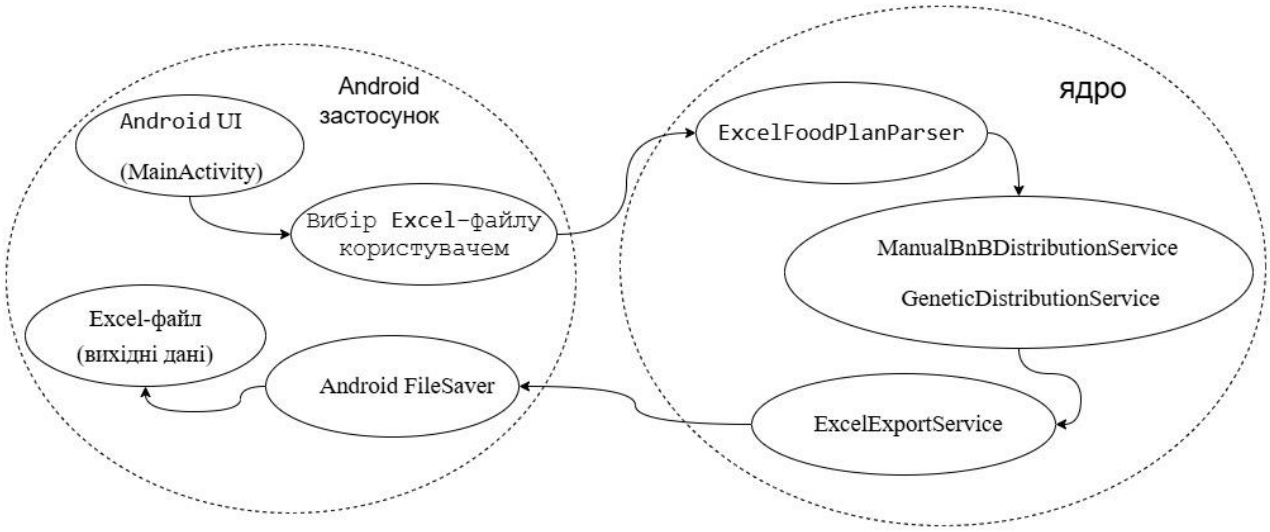


Рисунок 2.4 – Послідовність виконання сценарію в мобільній системі

Таке розмежування дозволяє підтримувати просту, легку реалізацію Android-застосунку при максимальній повторній придатності ядра.

2.2.3 Вибір технологій реалізації програмної системи

Вибір технологічного стеку для реалізації програмної системи FoodCalc Mobile визначався вимогами до продуктивності обчислень, надійності, можливості подальшого масштабування та переносимості бізнес-логіки між різними платформами. Оскільки система складається з двох взаємопов’язаних частин – ядра (foodcalc-mobile) та мобільного застосунку (foodcalc-android) – технологічні рішення повинні забезпечувати одночасно і незалежність, і сумісність між цими модулями. Тому основною платформою для реалізації ядра було обрано мову програмування Java версії 17, яка є довгостроковою (LTS) та містить сучасні можливості, що підвищують безпечність, читабельність та ефективність коду [21]. Java 17 забезпечує ефективне керування ресурсами, стабільність виконання та сумісність із сучасними бібліотеками для роботи з табличними й числовими обчисленнями. Крім того, застосування цієї версії мови гарантує сумісність ядра з різними операційними

системами та програмними платформами, що є важливим для подальшого розширення та повторного використання компонентів системи.

Організація збірки ядра виконується за допомогою системи керування проєктами Maven [22], що дозволяє підтримувати чітку структуру модулів, автоматично завантажувати залежності та створювати JAR-бібліотеку, яка інтегрується у мобільний застосунок. Вибір Maven є також наслідком архітектурної узгодженості із основним веб-проєктом FoodCalc, який використовує таку саму систему збірки. Це дозволяє підтримувати єдину інфраструктуру керування залежностями, спрощує подальшу інтеграцію між модулями та забезпечує керованість життєвого циклу проєкту. Maven, як класичний інструмент для створення Java-бібліотек, забезпечив передбачуваність збірки, відтворюваність результатів і стабільність у розробці ядра, яке виконує критичні обчислювальні функції.

Для роботи з файлами формату Excel, які є основним способом взаємодії користувачів із системою, у ядрі застосовано бібліотеку Apache POI. Цей інструмент дозволяє виконувати повний цикл операцій над файлами XLSX: читання, модифікацію, валідацію структури та формування підсумкових таблиць результатів розподілу. Вибір Apache POI був обґрунтований його зрілістю, активною підтримкою спільноти та сумісністю з Java 17. Окрім цього, у моделі предметної області використано бібліотеку Lombok, застосування якої дає змогу зменшити обсяг шаблонного коду та зосередити увагу на реалізації власне алгоритмічної логіки [23].

Окремого обґрунтування потребує вибір технологій для реалізації оптимізаційних алгоритмів. Для генетичного алгоритму було використано бібліотеку Jenetics версії 6.3.0, яка забезпечує потужний та гнучкий інструментарій для побудови еволюційних моделей. Jenetics підтримує формування генотипів, побудову функцій пристосованості, застосування операторів мутації та кросинговеру, а також паралельне виконання еволюційних обчислень, що є важливим для задачі розподілу пакунків. Використання цієї бібліотеки значно спростило створення якісної реалізації

генетичного алгоритму: потрібно було реалізувати лише специфічні для задачі механізми оцінювання рішень, тоді як еволюційний рушій забезпечується бібліотекою повністю. Водночас у процесі аналізу наявних бібліотек для методу гілок і меж в актуальних репозиторіях не знайдено стабільних, підтримуваних та достатньо гнучких рішень для Branch and Bound, які дозволяли б адаптувати алгоритм під специфіку багатоденного пакункового розподілу, реалізувати механізми раннього відсікання, контролювати дерево перебору та враховувати залежності між частинами пакунків. Тому алгоритм гілок і меж було реалізовано вручну, що дозволило повністю керувати структурою алгоритму й адаптувати його до вимог задачі.

Другим компонентом системи є мобільний застосунок foodcalc-android, який також використовує Java 17. Такий підхід забезпечує єдність технологічного стеку між ядром і клієнтом, дозволяє повторно використовувати логіку без дублювання коду та забезпечує коректну інтеграцію JAR-бібліотеки ядра на мобільному пристрої. Мобільний застосунок побудовано за допомогою Android SDK (мінімальна версія API 26, цільова версія API 36), а як систему збірки використано Gradle, оскільки Android Studio та офіційна екосистема Android повністю орієнтовані саме на цей інструмент. Gradle забезпечує інкрементальні збірки, гнучке конфігурування, високу продуктивність та підтримку сучасної архітектури Android-проєктів [24].

У проєкті застосовано сучасні AndroidX-компоненти, зокрема AppCompat, Material Components та ConstraintLayout, що забезпечує коректну роботу інтерфейсу на різних пристроях і підтримку сучасних стандартів дизайну [25], [26]. Робота із системними файловими діалогами реалізована за допомогою ActivityResult API, а для безпечної взаємодії з інтерфейсом використовується механізм ViewBinding. Оскільки система повинна працювати з Excel-файлами без підключення до сервера, у мобільному клієнті також застосовано Apache POI; таким чином обидва модулі – ядро та клієнт – працюють із однаковим форматом файлів і використовують єдині механізми для їх обробки. Інтеграція ядра з Android-застосунком реалізується через

підключення JAR-бібліотеки у директорію app/libs, що дає змогу виконувати всі обчислення локально та не залежати від зовнішньої інфраструктури.

Загалом обраний технологічний стек забезпечує високу продуктивність, модульність і довготривалу підтримку розробленої системи. Поєднання Java 17, Maven, Gradle, Jenetics, Apache POI та сучасних Android-компонентів дозволяє досягти стабільної роботи алгоритмів, забезпечити платформну незалежність ядра та реалізувати мобільний інтерфейс без дублювання логіки. Такий підхід створює гнучку та розширювану архітектуру, яка може бути адаптована для інших платформ і сценаріїв використання у майбутньому.

2.3 Реалізація алгоритму Branch and Bound

Задача, яку вирішує алгоритм, полягає у знаходженні такого розподілу харчових пакунків між учасниками багатоденного автономного маршруту, за якого кожен турист отримує навантаження з урахуванням його коефіцієнта навантаження. Навантаження кожного туриста має спадати монотонно з кожним днем за рахунок споживання продуктів. Відхилення від цільової (таргетної) ваги не повинно перевищувати ±10%. Кожен пакунок (набір продуктів) може використовуватися протягом кілька днів (наприклад, банка тушонки – 4 дні). Вага кількаденного пакунку розподіляється по днях відповідно до фактичного споживання, але додаткова вага (тара) додається лише в останній день його використання**.** Через мультиденну структуру пакунків модель має накопичувальний характер навантаження – вага частини пакунку, додана на початку розподілу, впливає на баланс усіх наступних днів. Тому клас PackageWithProducts містить набір денних частин пакунку, що дозволяє коректно враховувати внесок кожної частини пакунка у навантаження конкретного дня. Частина пакунка, додана у перший день його використання, впливає на баланс навантаження у всі наступні дні, у яких він має вагу. Тому

після першого призначення пакунка певному туристу всі його денні частини автоматично закріплюються за тим самим туристом. Відповідно, у разі відсікання певної гілки алгоритму виключаються всі частини цього пакунка одночасно, що забезпечує узгодженість перебору.

Реалізація алгоритму міститься у класі ManualBnBDistributionService пакету com.outdoor.foodcalc.domain.service. Код алгоритму BnB наведено у додатку В.

Загальну схему реалізації цього алгоритму подано на рисунку 2.5.

Основні сутності:

– HikerState;

– PackageWithProducts;

Основні методи:

– branchAndBound();

– assignPackagesOfDay();

– isFeasible();

– calculateTotalDeviation().

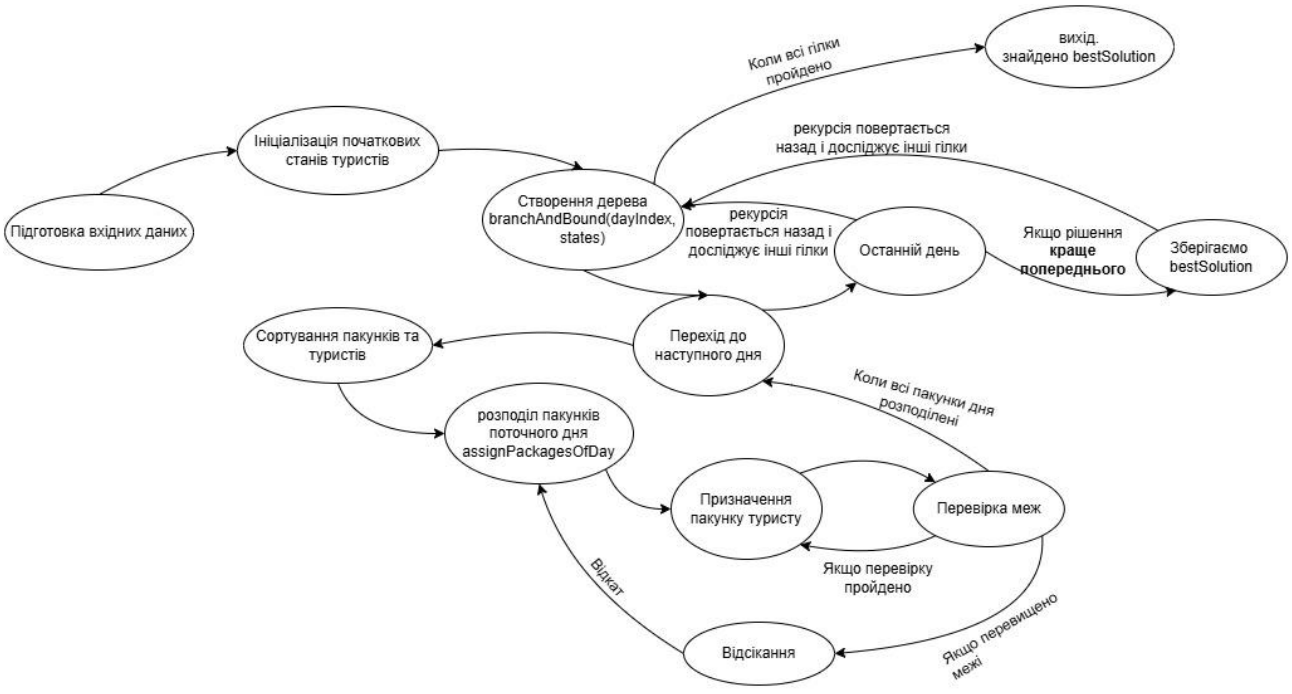


Рисунок 2.5 – Загальна схема реалізації алгоритму гілок і меж

HikerState – модель стану туриста, яка зберігає його навантаження за кожен день (loadByDay), призначені пакунки (assignedPackages), а також індивідуальні таргети (targetByDay).

PackageWithProducts – модель пакунка з розподілом ваги за днями (dayWeights), що визначає внесок пакунка у навантаження на кожну дату походу.

Метод branchAndBound() – рекурсивний механізм обходу дерева рішень, який визначає порядок розподілу пакунків між туристами.

Метод assignPackagesOfDay() – функція, яка здійснює призначення всіх пакунків конкретного дня та перевіряє допустимість навантаження після кожного кроку.

Метод isFeasible() – метод перевірки того, чи не перевищує навантаження туриста верхню межу (110% від індивідуального таргету) у дні, пов’язані з призначенням пакунка.

Метод calculateTotalDeviation() – фінальна метрика якості рішення, яка визначає середнє відхилення від таргетів для всіх туристів і днів.

2.3.1 Вхідні дані та підготовка

Перед запуском алгоритму виконується підготовка вхідних даних, необхідна для коректного формування дерева рішень. Спочатку за допомогою сервісу ExcelFoodPlanParser зчитується повний набір пакунків із вхідного Excel-файла, що містить дані для розподілу. Далі пакунки групуються за датами (prepareData()), у результаті чого формується впорядкована послідовність днів sortedDates, причому впорядкування здійснюється у зворотному хронологічному порядку – від останнього дня походу до першого. Після цього обчислюються групові таргети (calculateGroupTargets()), тобто сумарна вага всіх пакунків, які мають вагу у конкретний день. На основі цих значень визначаються індивідуальні таргети туристів (calculateIndividualTargets()), що враховують їхні коефіцієнти сили (weightCoefficient) і задають очікуване добове навантаження кожного учасника. Завершальним кроком є ініціалізація списку об’єктів HikerState, які відображають поточний стан туристів та містять

інформацію про їхнє денне навантаження, таргети та структуру призначених пакунків.

2.3.2 Структура дерева рішень

У методі гілок і меж процес пошуку ефективного розподілу відбувається шляхом рекурсивного будування дерева рішень. Це дерево відображає усі можливі комбінації призначення пакунків туристам із врахуванням послідовності днів та багатоденної структури ваги. Кожен рівень дерева відповідає одному дню маршруту, а глибина рекурсії визначається кількістю днів, для яких потрібно виконати розподіл.

Візуальне представлення побудови дерева рішень алгоритму подано на рисунку 2.6.

На кожному рівні рекурсії алгоритм формує підмножину пакунків, що мають вагу у поточний день (dayPackages). Ці пакунки впорядковуються за спаданням ваги, що дає змогу спершу розглядати найбільш ресурсомісткі комбінації. Такий підхід суттєво зменшує кількість неефективних гілок, оскільки важкі пакунки найімовірніше порушують межі припустимості, і невигідні комбінації відсікаються якомога раніше.

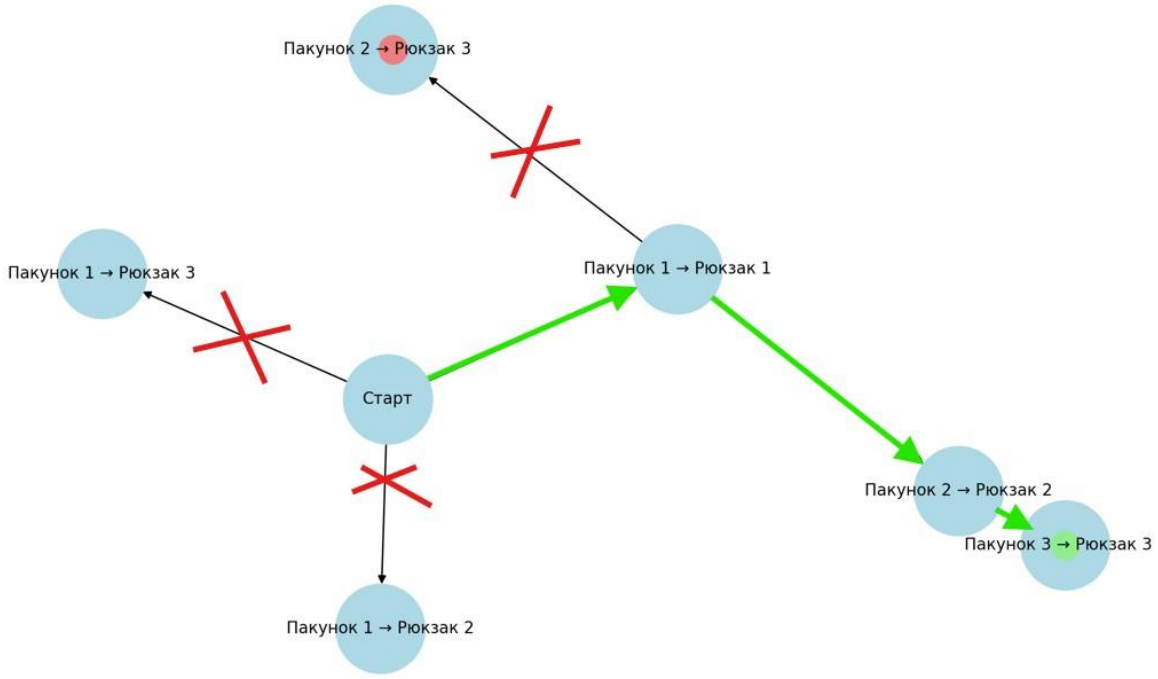


Рисунок 2.6 – Візуальна схема побудови дерева рішень алгоритму BnB

Паралельно з формуванням пакунків формується й порядок обходу туристів. На першому дні туристи сортуються за коефіцієнтом навантаження (weightCoefficient), щоб одразу встановити правильну пропорцію розподілу. На наступних днях порядок визначається сумарним фактичним навантаженням (getTotalWeightUpTo()), що дозволяє мінімізувати дисбаланс і зменшує кількість варіантів, які потребують розгляду. У такий спосіб алгоритм намагається призначити пакунки спочатку тим туристам, які мають найменше поточне навантаження.

Кожне призначення пакунка певному туристу породжує окрему гілку дерева рішень, у якій оновлюється стан конкретного туриста (навантаження за днями, призначені пакунки) та переходить розподіл на наступний пакунок поточного дня або до наступного дня загалом. Вершиною дерева є повністю сформоване рішення, у якому всі пакунки всіх днів призначені туристам, а для кінцевого варіанта здійснюється оцінка якості відповідно до метрики відхилення від таргетної ваги.

Завдяки рекурсивному принципу побудови дерево рішень охоплює увесь простір можливих комбінацій, а сортування та система відсікання дозволяють концентрувати обчислення на найбільш перспективних гілках. Така організація дає змогу ефективно поєднати повноту перебору із контрольованим відсіканням і забезпечує досягнення ефективного рішення у прийнятний час.

2.3.3 Межі (bound) та відсікання

Механізм відсікання у методі гілок і меж зменшує кількість варіантів перебору шляхом своєчасного припинення опрацювання тих гілок дерева рішень, які не можуть привести до допустимого або оптимального розподілу. У реалізованому алгоритмі використано два рівні контролю: локальне відсікання під час призначення пакунка та денне відсікання після завершення розподілу пакунків певного дня.

Першим рівнем є локальна перевірка у методі isFeasible(). Вона виконується одразу після додавання кожної частини пакунка до навантаження туриста. Перевіряється лише верхня межа – фактичне навантаження не повинно

перевищувати 110% від індивідуального таргету для відповідного дня. Якщо це обмеження порушено, подальший розподіл у цій гілці стає неможливим, і гілка негайно відсікається. Такий підхід дозволяє не виконувати зайвих обчислень для явно неефективних комбінацій.

Другим рівнем є перевірка після завершення призначення всіх пакунків конкретного дня, яка виконується у методі assignPackagesOfDay(). На цьому етапі алгоритм оцінює нижню межу припустимості: завантаження кожного туриста повинно становити не менше 90% від його таргету. Якщо хоча б один учасник отримав недостатнє навантаження, цей проміжний стан вважається неефективним, і алгоритм не переходить до розподілу пакунків наступного дня. Таким чином, гілка відсікається вже після повної оцінки дня.

Оскільки алгоритм працює з мультиденними пакунками, одним із важливих аспектів відсікання є цілісність пакунка. Після першого призначення пакунка певному туристу його вага додається на всі дні, у яких цей пакунок має споживання. Це забезпечує узгодженість моделі й запобігає ситуаціям, коли різні частини одного пакунка потрапили б до різних туристів. Відповідно, у разі відсікання гілки вилучаються всі денні частини пакунка, оскільки його призначення розглядається як неподільна операція.

Для підтримання коректності перебору кожна гілка працює із власною копією станів туристів. Метод cloneState() створює незалежні копії об’єктів HikerState, що гарантує повну ізольованість гілок дерева рішень. Завдяки цьому відсікання однієї гілки не впливає на інші можливі комбінації.

Такий підхід до механізму меж і відсікання забезпечує баланс між повнотою перебору та ефективністю роботи алгоритму, значно скорочуючи кількість неефективних варіантів та підвищуючи швидкість пошуку розв’язання.

2.3.4 Вибір найкращого рішення

У методі гілок і меж передбачається не лише пошук будь-якого допустимого розподілу пакунків, а й визначення найбільш ефективного з усіх можливих варіантів. Тому після кожного успішного проходження всіх днів та

повного формування проміжного розподілу алгоритм виконує обчислення його якісної метрики – середнього відхилення фактичного навантаження від таргетних значень для всіх туристів та всіх днів.

Оцінювання здійснюється у методі calculateTotalDeviation(), який для кожного туриста та кожного дня визначає відносне відхилення (2.1):

$$deviation_{jd} = \frac{|L_{jd} - W_{jd}|}{W_{jd}}, \quad (2.1)$$

де j – турист;

d – день;

L_{jd} – фактичне навантаження туриста в день d;

W_{jd} – цільова вага для туриста в день d.

Після цього обчислюється середнє значення відхилення для всіх комбінацій «турист – день», яке і є підсумковим показником якості рішення. Чим меншим є це значення, тим рівномірніший та збалансованіший розподіл отримали туристи.

Алгоритм зберігає поточне найкраще знайдене рішення у спеціальній структурі bestSolution. Під час перебору кожне нове допустиме рішення порівнюється з уже наявним найкращим. Якщо його метрика є кращою (тобто менша), поточний розподіл зберігається як новий bestSolution. Для запобігання взаємному впливу різних гілок стан поточного рішення клонуються перед додаванням до bestSolution, що гарантує його правильність та незалежність.

Такий підхід дозволяє алгоритму не лише визначити розподіл, який задовольняє всі обмеження, але й вибрати той варіант, який забезпечує найменше сумарне відхилення від цільових ваг. Це відповідає загальній меті задачі – досягти максимально рівномірного навантаження між учасниками протягом усього маршруту і мінімізувати ризик перевантаження чи недовантаження окремих туристів.

2.3.5 Результати виконання та складність

Після повного обходу дерева рішень і завершення рекурсивного перебору алгоритм формує підсумковий найкращий розподіл, який зберігається у структурі bestSolution. Ця структура містить повний набір об’єктів HikerState для кожного туриста, включно з інформацією про денне навантаження, призначені пакунки та відсоткові відхилення від цільових значень. Такий формат представлення результатів дозволяє виконати подальший аналіз розподілу та експорт сформованих даних у зручні для користувача формати. Для формування звіту застосовується модуль експорту Excel, який створює вихідний файл із докладною аналітичною інформацією. Зокрема, у листі «Details» відображається:

– цільова вага (Target For Day) для кожного туриста і кожного дня;

– фактичне навантаження (Assigned Weight For Day), отримане за результатами роботи алгоритму;

– відносне відхилення у відсотках, що характеризує точність досягнення цільового значення;

– перелік призначених пакунків, які формують це навантаження.

Ці дані дозволяють оцінити якість рішення, порівняти учасників між собою і візуально перевірити коректність роботи алгоритму.

Метод гілок і меж у загальному випадку має експоненційну обчислювальну складність у найгіршому випадку $O(m^n)$, де m – кількість туристів; n – кількість пакунків. Такий сценарій відповідає випадку, коли кожен пакунок може бути призначений будь-якому учаснику, а відсікання майже не відбувається. Проте фактична складність реалізованого алгоритму є значно нижчою завдяки використанню низки оптимізацій. Зокрема, попереднє сортування пакунків за спаданням ваги дає змогу швидше усувати неефективні комбінації, тоді як впорядкування туристів за поточним навантаженням або коефіцієнтом зменшує дисбаланс на ранніх етапах розподілу. Додатково застосовується локальна перевірка допустимості (isFeasible()), яка негайно припиняє розвиток гілки за порушення обмежень, а також глобальна перевірка

після завершення кожного дня, що блокує подальший перебір несумісних розв’язків. Окрему роль відіграє механізм клонування станів (cloneState()), який запобігає накопиченню помилок і забезпечує незалежність гілок дерева рішень.

У комплексі такі заходи істотно звужують простір пошуку та дозволяють отримувати рішення у прийнятний час для реалістичних сценаріїв, що містять десятки пакунків і кілька учасників. Це підтверджує, що практична продуктивність алгоритму значно краща за теоретичні оцінки найгіршого випадку, а сам метод гілок і меж є ефективним інструментом для задачі розподілу ваги в автономних походах.

2.4 Реалізація алгоритму Genetic Algorithm

Генетичний алгоритм у системі FoodCalc Mobile використовується як альтернативний евристичний метод для оптимізації розподілу багатоденних харчових пакунків між туристами. Мета його роботи аналогічна до задачі, що розв’язується методом гілок і меж: забезпечити таке призначення пакунків, за якого щоденне навантаження кожного туриста максимально наближене до його індивідуального таргету, обчисленого пропорційно коефіцієнту сили. Відхилення від таргету не повинно перевищувати ±10% у будь-який день походу. Особливістю задачі є багатоденність пакунків, тобто пакунок може мати кілька денних частин, і додаткова вага тари додається лише в останній день його використання.

Генетичний алгоритм застосовується у тих випадках, коли простір пошуку є надто великим для точних методів або коли потрібна швидка евристична оцінка. GA забезпечує наближення до оптимального розподілу за прийнятний час, використовуючи механізми еволюції: формування поколінь, стохастичний добір, мутації та кросинговер. У реалізованому підході

використано бібліотеку Jenetics, що забезпечує повний інструментарій для побудови генотипів, функцій пристосованості та еволюційного рушія.

Загальна ідея алгоритму полягає в представленні розподілу пакунків у вигляді хромосоми, де кожен ген відповідає пакунку, а його значення визначає туриста, якому цей пакунок призначено [18]. Еволюційний цикл поступово покращує це призначення, мінімізуючи сумарне відхилення від таргетної ваги та дотримуючись обмежень задачі.

Реалізація генетичного алгоритму зосереджена в класі GeneticDistributionService (пакет com.outdoor.foodcalc.domain.service). Цей сервіс відповідає за формування еволюційної моделі, запуск еволюційного циклу та побудову фінального рішення. Архітектура алгоритму побудована відповідно до принципів Jenetics і включає декілька етапів: підготовка вхідних даних, створення генотипу, визначення функції оцінки, налаштування параметрів еволюції та запуск еволюційного двигуна.

У реалізації використовуються такі основні сутності:

– PackageWithProducts;

– HikerState.

PackageWithProducts – модель пакунка з розподілом ваги за днями. Денна вага зберігається у структурі dayWeights. Метод getWeightForDay(day, members) забезпечує коректне визначення частки ваги для конкретного туриста та конкретного дня, що є важливим для розрахунку навантаження під час оцінки хромосоми.

HikerState – стан туриста, який фіксує навантаження за всі дні (loadByDay), індивідуальні таргети (targetByDay) та список призначених пакунків. У циклі оцінки цей клас використовується для відновлення фактичного навантаження туриста на основі хромосоми та для подальшого аналізу відхилень.

Модуль експорту результатів ExcelExportService формує звіт за двома ключовими аркушами: «Distribution» (загальний розподіл пакунків) та «Details» (порівняння таргету і фактичного навантаження). Цей сервіс дозволяє

візуалізувати результат роботи GA у форматі Excel і забезпечує однаковий стиль формування звітів для GA та BnB.

Архітектура реалізації генетичного алгоритму передбачає послідовне виконання низки етапів. На початковому етапі здійснюється ініціалізація даних, що включає формування списків туристів, пакунків та індивідуальних таргетів. Після цього створюється генотип: у реалізації використовується структура IntegerGene, де кожен ген представляє індекс туриста, якому призначено певний пакунок. Далі визначається функція оцінки, яка обчислює сумарне відхилення від цільових значень і таким чином задає критерій якості для кожної хромосоми. Після формування функції пристосованості задаються параметри еволюції, включаючи розмір популяції, кількість поколінь, ймовірності мутації та кросинговеру, а також правила селекції. На основі цих параметрів створюється та запускається еволюційний двигун, який поступово генерує покоління хромосом і відбирає серед них кращі варіанти, наближаючи рішення до оптимального. Після завершення еволюційного процесу з найкращого знайденого фенотипу будується фінальний розподіл, який і передається модулю експорту для формування вихідного Excel-звіту.

Така узгоджена послідовність дій формує повний цикл роботи алгоритму, що відображено на структурній схемі на рисунку 2.7.

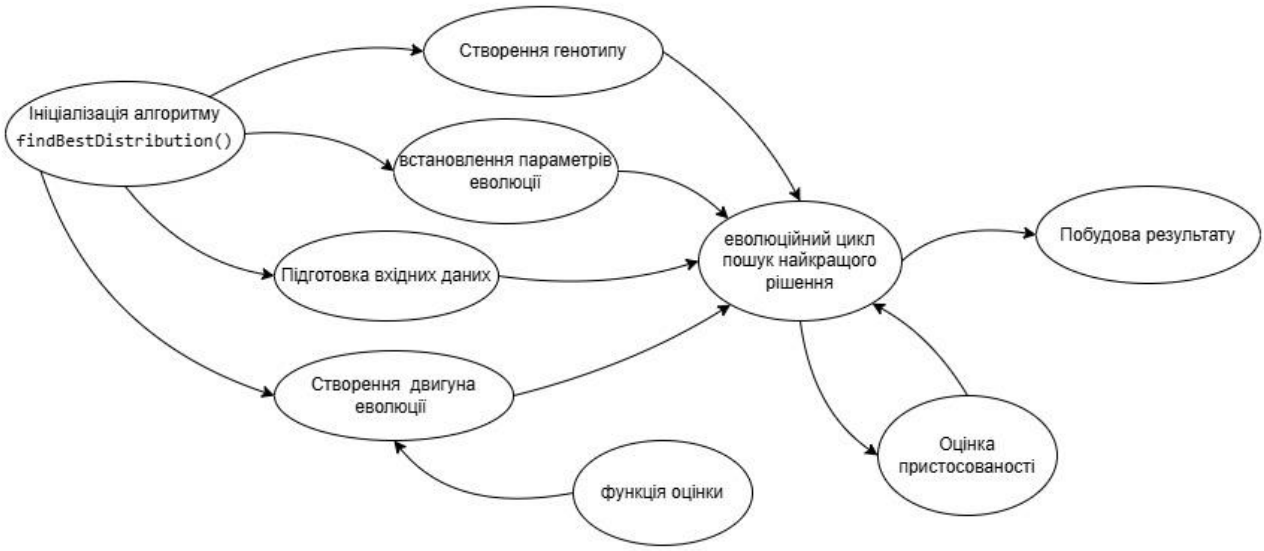


Рисунок 2.7 – Структурна схема архітектури реалізації генетичного алгоритму

Еволюційний цикл дає змогу наближатися до ефективного розподілу навіть у великих просторах пошуку, де метод гілок і меж був би надто повільним або обчислювально непридатним.

2.4.1 Кодування рішення (генотип)

Ефективність роботи генетичного алгоритму значною мірою залежить від способу кодування розв'язку, оскільки саме генотип визначає структуру простору пошуку та можливість застосування операторів еволюції [18]. У реалізованому підході розподіл пакунків між туристами подано у вигляді одномірної хромосоми фіксованої довжини, де кожен ген кодує вибір туриста для відповідного пакунка.

Хромосома являє собою масив генів довжини n, де n – загальна кількість пакунків у плані харчування. Кожен ген є цілим числом у діапазоні $[0, m - 1]$, що відповідає індексу туриста, якому призначається відповідний пакунок. Таке кодування є природним для задачі, оскільки кожен пакунок повинен бути закріплений за рівно одним туристом, а сам генотип залишається компактним і зручним для застосування операторів мутації та кросинговеру.

Після завершення еволюційного циклу Jenetics перетворює найкращу хромосому у фенотип. На цьому етапі хромосома трансформується у структури доменної моделі – зокрема у список об'єктів HikerState. Для кожного гена розподіл пакунків відтворюється таким чином, що туристу, якому призначено відповідний пакунок, додаються всі його денні частини у відповідні дні походу. Таким чином формується повна модель фактичного навантаження для кожного туриста, яка надалі використовується для обчислення таргетів, відхилень і формування фінального рішення.

Обране представлення є узгодженим із структурою вхідних даних та дозволяє забезпечити коректну роботу функції пристосованості, оскільки всі необхідні параметри (розподіл ваги за днями, структура пакунків і коефіцієнти туристів) можуть бути відтворені безпосередньо з генотипу. Такий підхід робить кодування рішення компактним, інформативним та придатним для операцій еволюційного пошуку.

2.4.2 Таргети (цільові ваги)

Цільові ваги (таргети) визначають еталонне навантаження, яке кожен турист має нести щодня відповідно до своїх можливостей (weightCoefficient). Розрахунок виконується у два кроки. Спочатку для кожного дня обчислюється групова вага L_d – сумарна маса всіх пакунків цього дня з урахуванням множника учасників, коефіцієнтів об'єму та ваги тари, що додається лише в останній день використання пакунка.

Далі ця групова вага розподіляється між туристами пропорційно їхнім коефіцієнтам за формулою (1.9).

Таким чином формується набір індивідуальних таргетів, який використовується під час оцінки хромосом та для формування Excel-звіту. Значення зберігаються у HikerState.targetByDay і надалі порівнюються з фактичним навантаженням для оцінки якості розподілу.

2.4.3 Функція пристосованості (fitness)

Функція пристосованості визначає якість розподілу та керує еволюційним процесом, спрямовуючи пошук до рішень, у яких щоденні навантаження туристів максимально відповідають їхнім індивідуальним таргетам [18]. Базовим елементом оцінювання є відносне відхилення (формула 2.1), яке показує, наскільки фактична вага для туриста в певний день перевищує або недосягає таргетного значення. На основі цього відхилення формується трикомпонентна вартість розв'язку: денний жорсткий штраф, денний м'який штраф та підсумковий штраф за весь маршрут. У сукупності вони формують величини *dailyOutside*, *dailyInside* та *tripTotals*, що входять у формулу загальної вартості (2.2).

Загальна вартість розв'язку обчислюється як лінійна комбінація трьох складових:

$$cost = A \cdot dailyOutside + B \cdot dailyInside + D \cdot tripTotals, \quad (2.2)$$

Перша складова, *dailyOutside*, відповідає за випадки, коли абсолютне відхилення від таргету виходить за межі допустимого діапазону ±10 %. Якщо у певний день фактичне навантаження туриста перевищує цей поріг, відповідна величина відхилення вносить у загальну вартість так званий жорсткий денний штраф. Він зростає кубічно $(deviation_{jd} - 0.10)^3$ залежно від величини перевищення, що забезпечує різке покарання рішень , у яких навантаження вийшло за рамки встановленого добового допуску. Саме ця складова є домінантною у (2.2) – коефіцієнт A забезпечує пріоритетну важливість дотримання щоденного допустимого інтервалу.

Друга складова, *dailyInside*, охоплює випадки, коли відхилення туриста від денного таргету все ще знаходиться в межах допустимих ±10 %, але не є рівним нулю. У ситуаціях коли $deviation_{jd} \leq 0.10$ накладається м'який денний штраф , який зростає плавно зі збільшенням відхилення, але не обмежує пошук і не відсікає потенційно перспективні хромосоми. Ця частина відповідає за те, щоб алгоритм прагнув більш рівномірних та точних розподілів там, де це можливо, але при цьому не втратив здатності досліджувати різноманітні варіанти. Внесок цієї складової у формулу (2.2) контролюється коефіцієнтом *B*.

Третя складова, *tripTotals*, визначає якість розподілу на рівні всього маршруту. Навіть якщо у кожен окремий день турист утримувався в межах допустимого відхилення, його накопичене навантаження за всі дні може виявитися нерівномірним відносно його загального таргетного навантаження. Для цього порівнюється сумарна фактична вага, яку несе турист протягом усього походу, із сумою його денних таргетів. Відхилення до ±5 % не штрафується, оскільки враховує природну неточність під час багатоденного планування; однак перевищення цього порогу формує підсумковий штраф, який збільшується квадратично. Таким чином ця складова забезпечує збалансованість між туристами у масштабі всього маршруту, а її вплив задається коефіцієнтом *D* у формулі (2.2).

Коефіцієнти *A*, *B* та *D* підібрані так, щоб головним критерієм була відсутність перевищення добового порогу ±10 %, додатковим – точність

усередині допуску, та завершальним – справедливість загального балансу між туристами. Після обчислення вартості виконується нормалізація за формулою (2.3), яка перетворює її у значення fitness: що менша вартість, то ближчий розподіл до оптимального та тим вищим є fitness. Фітнес визначається за стандартною нормалізацією (2.3):

$$fitness = \frac{1}{1+cost}. \quad (2.3)$$

Така форма нормалізації гарантує коректну роботу еволюційного механізму Jenetics, оскільки максимальне значення fitness відповідає найкращим розв'язкам.

2.4.4 Еволюційний двигун і параметри

Еволюційний процес у генетичному алгоритмі реалізовано за допомогою рушія Jenetics, який формує послідовність поколінь та керує застосуванням операторів еволюції. Конфігурація еволюційного двигуна визначає баланс між стабільністю пошуку та різноманітністю рішень, що безпосередньо впливає на швидкість і якість збіжності.

У реалізованій моделі використано популяцію приблизно з 400 особин, що забезпечує достатню варіативність хромосом при помірних обчислювальних витратах. До хромосом застосовуються декілька операторів еволюції: базова мутація з імовірністю 25%, додаткова swap-мутація з імовірністю 5%, а також оператор кросинговеру з коефіцієнтом 0.6, який забезпечує змішування генетичного матеріалу різних особин. Така комбінація дозволяє поєднати локальні зміни з глобальною перебудовою структури розподілу.

Для збереження якісних рішень використовується елітизм, який гарантує перехід трьох найкращих особин до наступного покоління без змін. Частка нащадків встановлена на рівні 0.7, завдяки чому більша частина нового покоління формується із застосуванням операторів еволюції, що сприяє швидшому покращенню популяції.

Еволюційний цикл триває приблизно 320 поколінь, і процес оптимізації спрямований на максимізацію значення fitness. Усі наведені параметри були визначені емпірично: у ході тестування вони забезпечили стабільне наближення до рішень, у яких щоденні відхилення фактичного навантаження від таргету переважно утримуються в межах допустимих ±10%.

2.4.5 Побудова результату та складність

Після завершення еволюційного процесу найкращий фенотип, отриманий з еволюційного рушія Jenetics, трансформується у структури доменної моделі. Хромосома перетворюється у список об’єктів HikerState, у яких для кожного туриста формується послідовність денних навантажень та фіксуються призначені пакунки. Для кожного дня зберігається фактична маса, яка відповідає сумі всіх денних частин пакунків, призначених туристу, із врахуванням ваги тари в останній день використання. Ці дані використовуються для формування Excel-звіту, зокрема аркуша «Details», де подаються показники Target For Day, Assigned Weight For Day, Deviation (%) та перелік отриманих пакунків.

Генетичний алгоритм не гарантує глобальний оптимум, зате добре масштабується: замість експоненційного перебору GA шукає наближене розв’язання у великому просторі рішень, балансуючи експлорацію (мутації) і експлуатацію (кросовер та елітизм). Час виконання і якість результату регулюються розміром популяції, кількістю поколінь і силою штрафів у fitness.

Обчислювальна складність алгоритму визначається параметрами еволюції: розміром популяції, кількістю поколінь та ваговими коефіцієнтами у функції пристосованості. Збільшення цих параметрів підвищує якість рішення, однак збільшує час виконання. Таким чином, GA надає гнучкий механізм керування балансом між точністю та швидкістю, що робить його придатним для задач багатоденного розподілу з великим числом варіантів.

2.5 Висновки до розділу

У другому розділі розроблено архітектуру програмної системи та описано алгоритмічні підходи до вирішення задачі. Сформовано доменну модель, яка відображає ключові сутності предметної області – туристів, дні походу, пакунки та їх багатоденну структуру використання. Визначено механізми обчислення цільових навантажень, контролю монотонного спадання ваги та врахування індивідуальних коефіцієнтів навантаження. Запропонована архітектура ядра та мобільного застосунку забезпечує розділення відповідальності, повторне використання бізнес-логіки, сумісність між модулями та можливість автономної роботи на мобільних пристроях. У межах розділу побудовано дві самостійні реалізації алгоритмів розв’язання задачі – точного та еволюційного типу, – для кожного з яких визначено структуру даних, процес формування рішення та механізми оцінювання якості розподілу. Створені алгоритмічні моделі та їх інтеграція в архітектуру системи формують підґрунтя для подальших експериментальних досліджень, спрямованих на аналіз поведінки, обчислювальних характеристик і придатності реалізованих методів до практичного використання для планування автономних походів.

3 ПРОВЕДЕННЯ ЕКСПЕРИМЕНТАЛЬНИХ ДОСЛІДЖЕНЬ

3.1 Формальні критерії адекватності моделі

Адекватність моделі оцінюється за сукупністю кількісних критеріїв, що характеризують точність, стабільність та ефективність отриманих розв’язків, а також за якісними ознаками відповідності реальним умовам використання мобільної системи планування навантажень. Одним із ключових показників є добовий баланс навантаження, який визначається відносним відхиленням між

фактичним навантаженням туриста та його цільовою вагою за формулою (2.1). Вважається прийнятним, якщо відхилення не перевищує допустиме відхилення σ [15]. Виконання цієї умови гарантує рівномірний розподіл фізичного навантаження протягом усього маршруту та відображає базову межу прийнятності розв'язання.

Додатковою умовою коректності є монотонне спадання ваги рюкзака, зумовлене поступовим споживанням продуктів [9]. Для будь-якого туриста j навантаження в наступний день походу $d+1$ має бути меншим або рівним навантаженню в день d, тобто $W_{j,d+1} \leq W_{jd}$. Порушення цієї властивості свідчить про некоректний розподіл або повторне врахування частин тих самих пакунків.

Суттєвим елементом моделі є також пропорційність навантаження відповідно до коефіцієнтів туристів. Сумарна вага, яку несе кожен учасник, має відповідати його частці в загальній груповій масі маршруту, тобто співвідношення навантажень між туристами повинно узгоджуватися з їхніми коефіцієнтами сили. Це забезпечує фізіологічно виправданий баланс і відображає реальні можливості учасників.

Окреме значення має цілісність багатоденних пакунків. Усі денні частини одного пакунка мають бути призначені одному туристу, що унеможливлює ситуації, за яких окремі фрагменти розподіляються між різними учасниками. Така вимога є принципово важливою для коректності моделі, оскільки пакунок у реальному поході транспортується одним рюкзаком незалежно від багатоденної структури його ваги [9].

Адекватність моделі оцінюється також за критерієм обчислювальної ефективності. Для типового сценарію, що включає п'ять туристів, сім днів і близько п'ятдесяти пакунків, час розв'язання задачі має залишатися в межах, придатних для використання на мобільних пристроях. Зокрема, час виконання генетичного алгоритму повинен становити не більше кількох секунд, тоді як для методу гілок і меж допустимим є виконання в межах однієї–двох хвилин у

випадках повного перебору. Це гарантує практичну застосовність моделі в умовах, коли обчислювальні ресурси користувача обмежені.

Важливим критерієм є відтворюваність результатів. Для детермінованого методу гілок і меж відтворюваність забезпечується самою природою алгоритму. Для стохастичного генетичного алгоритму її досягають за допомогою фіксації початкового зерна генератора випадкових чисел. Девіація результатів між запусками для контрольних наборів даних не повинна перевищувати двох відсотків [17].

Ще одним важливим елементом оцінювання є масштабованість. Модель вважається адекватною, якщо зі збільшенням кількості туристів або пакунків у два рази час виконання не зростає більш ніж у чотири рази, а середні відхилення добового навантаження залишаються в межах встановленої норми ±10 %. Така поведінка свідчить про стійкість алгоритмів до розширення задачі та підтверджує придатність моделі для застосування у ширшому спектрі сценаріїв.

3.2 Порівняльна характеристика алгоритмів

Два реалізовані алгоритми – BnB та GA – належать до різних класів методів оптимізації. BnB – детермінований точний метод, що гарантує глобальний оптимум [7]. GA – стохастична метаевристика, яка шукає наближене рішення в прийнятний час [18]. Основні відмінності наведено в таблиці 3.1.

Для оцінювання стабільності роботи алгоритмів проведено аналіз їхньої чутливості до зміни параметрів задачі. Основні спостереження наведено в таблиці 3.2.

Таблиця 3.1 – Порівняльна характеристика алгоритмів BnB та GA

Ознака	BnB	GA
Тип методу	Точний, перебірний	Стохастичний, еволюційний
Гарантія оптимуму	Так	Ні (наближене)
Складність	Експоненційна	Лінійна
Вимоги до пам’яті	Високі	Помірні
Час виконання	Зростає експоненційно з m і n	Керується параметрами λ та G
Відтворюваність	Повна	Імовнісна (через seed)
Придатність	Малі та середні задачі	Великі та динамічні задачі
Можливість гібридизації	Висока	Висока

Таблиця 3.2 – Чутливість алгоритмів BnB та GA до параметрів задачі

Параметр	Вплив на BnB	Вплив на GA
Допуск σ (±10%)	При зменшенні σ дерево пошуку розширюється, оскільки різко зменшується кількість допустимих проміжних станів, що збільшує обсяг перебору.	При зменшенні σ час виконання зростає лінійно, GA потребує більше поколінь або більшу популяцію.
Кількість днів D	Збільшує глибину рекурсії.	Час виконання зростає лінійно, зростає час обчислення fitness $O(n \cdot D)$
Кількість туристів m	Збільшення m веде до мультиплікативного зростання, кожен пакунок породжує m можливих гілок.	Час зростає лінійно, збільшується кількість альтернатив для кожного гену, fitness стає важчим для стабілізації
Кількість пакунків n	Найкритичніший параметр, дає найбільший внесок у експоненційну складність $O(m^n)$.	Час збільшується майже пропорційно n.

Оцінки впливу параметрів базуються як на теоретичних властивостях алгоритмів, так і на спостереженнях, отриманих у ході запусків реалізованих модулів ManualBnBDistributionService та GeneticDistributionService.

3.3 Оцінювання поведінки моделі на реалістичних сценаріях

Запропонований модельний підхід демонструє відповідність реалістичним умовам багатоденного автономного походу, оскільки враховує ключові фізіологічні та логістичні обмеження, характерні для розподілу ваги

між учасниками. Обраний допуск відхилення навантаження на рівні ±10% є практично обґрунтованим: він відповідає типовим межам дискомфорту під час перенесення спорядження та дозволяє забезпечити баланс між точністю моделі та можливістю досягнення розв’язку в прийнятний час. Проведені спостереження засвідчили, що за умови використання задач реалістичних масштабів модель стабільно забезпечує дотримання вимог добового балансу, пропорційності навантаження учасників та монотонного спадання ваги рюкзаків. Це підтверджує адекватність обраного підходу та його придатність до подальшої практичної реалізації в мобільній системі планування рівномірного розподілу харчових пакунків у багатоденних туристичних походах.

3.4 Межі застосування

Ефективність застосування розроблених алгоритмів залежить від масштабів задачі та комбінації параметрів, зокрема кількості туристів, тривалості маршруту й обсягу пакунків. Метод гілок і меж доцільно використовувати для задач малих та середніх розмірів, коли кількість учасників не перевищує приблизно 5–7 осіб, а загальна кількість пакунків становить до 30–40 одиниць. У таких умовах алгоритм здатний забезпечити точний результат із гарантованим перебором, зберігаючи прийнятний час обчислень [7]. Для масштабніших задач, що охоплюють до десяти туристів, тривалість маршруту понад десять днів і сотню пакунків, більш придатним є генетичний алгоритм, оскільки він здатний генерувати якісні наближені рішення без експоненційного зростання часу виконання [18]. Для задач, у яких значення параметрів m, D або n суттєво відрізняються від базових умов, рекомендовано виконувати окреме калібрування параметрів генетичного алгоритму, зокрема розміру популяції, кількості поколінь та інтенсивності

операторів мутації й кросинговеру. Такий підхід забезпечує стабільність та надійність моделі в різних масштабах задачі.

Таким чином, кожен із підходів має власні межі ефективності, а вибір алгоритму залежить від розмірності вхідних даних та допустимих обмежень щодо часу виконання.

3.5 Мета та умови експериментів

Експериментальні дослідження проведено з метою оцінити ефективність реалізованих у ядрі системи алгоритмів BnB та GA у контексті задачі розподілу багатоденних харчових пакунків між туристами. Головною метою є визначення того, як розмір задачі, характер пакунків та обрані параметри алгоритмів впливають на точність денного балансування навантаження, стабільність результатів та час обчислень, що є критично важливим для подальшого використання обчислювальної логіки на мобільних пристроях.

Усі експерименти виконувалися безпосередньо на основі модулів ядра foodcalc-mobile, де реалізовано два незалежні підходи до розв’язання задачі:

– ManualBnBDistributionService – реалізація детермінованого методу BnB;

– GeneticDistributionService – реалізація еволюційного алгоритму GA.

Підготовка вхідних даних для обох алгоритмів здійснювалася за допомогою модуля ExcelFoodPlanParser, який формує структуру FoodPlan на основі завантаженого Excel-файлу плану харчування. Після завершення обчислень результати зберігалися засобами у форматі XLSX з формуванням листів «Distribution» та «Details». Ці листи містять інформацію про фактичний розподіл пакунків, денні навантаження туристів, відхилення від цільових значень та структуру пакунків.

Умови експериментів передбачали використання наборів планів різної складності – від малих сценаріїв (3–4 туристи, до 20 пакунків) до розширених,

орієнтованих на мобільне застосування (5–7 туристів, 40–60 пакунків). Це дало змогу оцінити поведінку обох алгоритмів як за точністю денного балансування та відповідності таргетам, так і за обчислювальною придатністю до використання на пристроях із обмеженими ресурсами.

3.6 Апаратні ресурси дослідження

Апаратна складова дослідження охоплює як характеристику категорії пристроїв, для яких призначена мобільна система FoodCalc Mobile, так і конкретні технічні параметри обладнання, на якому проводилися експериментальні випробування. Застосунок орієнтований на роботу на персональних Android-пристроях масового сегмента, що використовують процесори архітектури ARM і забезпечують роботу середовища Android Runtime у поєднанні з ядром Java 17. Для коректної роботи достатньо обчислювальних ресурсів рівня багатоядерного процесора ARM Cortex-A53 або Qualcomm Snapdragon початкових серій, а також обсягу оперативної пам’яті від 2–4 ГБ, що гарантує можливість виконання алгоритмів у реальному часі без залучення зовнішніх обчислювальних сервісів. Застосунок не потребує мережевого з’єднання й функціонує повністю автономно, а обсяг вільної пам’яті близько 50 МБ є достатнім для розміщення ядра системи, вхідних файлів і результатів експорту.

Експериментальні випробування виконувалися на смартфоні середнього класу Xiaomi Redmi Note 10 під керуванням Android 12, з процесором Qualcomm Snapdragon 678 з конфігурацією ядер 2 × 2.2 GHz і 6 × 1.7 GHz та 4 ГБ оперативної пам’яті. Таке апаратне забезпечення репрезентує поширений сегмент мобільних пристроїв, доступних більшості користувачів, і дозволяє оцінити працездатність системи без використання спеціалізованого обладнання. Під час експериментів забезпечувався контроль умов виконання:

пристрій працював без додаткових навантажувальних процесів та фонових служб, які могли б впливати на час обчислень; температура процесора залишалася стабільною, що мінімізувало ризик зниження продуктивності через перегрів. Таким чином створено однакові апаратні умови для всіх запусків алгоритмів, що забезпечило відтворюваність і коректність експериментальних результатів [27].

3.7 Конфігурація програмного середовища

Для забезпечення відтворюваності експериментів та коректного порівняння алгоритмів у мобільному середовищі було зафіксовано програмні параметри, на яких здійснювалося тестування. Обчислення виконувалися вбудованим Java-ядром системи FoodCalc Mobile, реалізованим у вигляді JAR-бібліотеки, що працює у середовищі Android Runtime.

Експерименти проводилися у стабільному та контрольованому програмному оточенні: на пристрої були вимкнені всі несуттєві служби, а фонові процеси мінімізовані, що дозволило уникнути впливу зовнішніх факторів на результати. Для вимірювання часу виконання застосовувався внутрішній високоточний таймер, побудований на основі функції System.nanoTime(), інтегрований безпосередньо у методи розподілу. Це забезпечило точність вимірювань та їх незалежність від інструментів операційної системи або стану інтерфейсу. Усі експерименти проводилися послідовно в обох реалізаціях (Branch and Bound та Genetic Algorithm) в однакових умовах, що мінімізувало можливі спотворення результатів, пов’язані з фоновими навантаженнями, змінами системних ресурсів або поведінкою Android-сервісів. Такий підхід дозволяє вважати отримані результати коректними для порівняльного аналізу та достатніми для оцінювання поведінки алгоритмів у мобільному середовищі.

3.8 Набори вхідних даних та параметри алгоритмів

Для проведення експериментів було сформовано три набори вхідних даних, які відрізняються кількістю туристів, тривалістю походу та кількістю пакунків. Характеристики кожного набору наведено в таблиці 3.3. Детальні таблиці використання денних частин пакунків для всіх наборів вхідних даних наведено в додатку А.

Таблиця 3.3 – Набори вхідних даних для експериментів

№	Назва набору	Опис	Розмір
1	TestPlan4x4	4 туристи × 4 дні, 21 пакунок	53 денних частин пакунків
2	Тянь-Шань (великі пакунки)	3 туристи × 3.5 дні, 10 пакунків	38 денних частин пакунків
3	Тянь-Шань (дрібні пакунки)	модифікований набір 3 туристи × 3.5 дні, 14 пакунків	44 денних частин пакунків

Для кожного з реалізованих алгоритмів визначено набір параметрів, що впливають на процес розподілу пакунків між туристами. Характеристики параметрів алгоритму BnB наведено в таблиці 3.4, а генетичного алгоритму GA – у таблиці 3.5.

Таблиця 3.4 – Параметри алгоритму BnB (ManualBnBDistributionService)

Параметр	Значення	Призначення
Сортування пакунків	за спаданням	Мінімізує кількість неперспективних гілок
Сортування туристів (перший день)	за коефіцієнтом сили	Забезпечує пропорційність навантаження
Сортування туристів (наступні дні)	за фактичним навантаженням	Балансує вагу під час ітерацій
Допустиме відхилення	±10 %	Відсікаються гілки, де порушено допуск
Монотонність спадання ваги	обов’язкова	Навантаження повинно спадати по днях
Цілісність пакунків	обов’язкова	Усі частини пакунка належать одному туристу
Критерій оптимальності	мінімум середнього відхилення по вазі для кожного туриста	Забезпечує рівномірність розподілу
Вихід	структура HikerState	Містить розподілену вагу, цільову вагу, пакунки кожного туриста

Таблиця 3.5 – Параметри алгоритму GA (GeneticDistributionService)

Параметр	Значення	Призначення
Тип гена	IntegerGene (0…m−1)	Індекс туриста для кожного пакунка
Розмір популяції (λ)	400	Кількість рішень у поколінні
Кількість поколінь (G)	320	Межа еволюції
Мутація (p_m)	0.25	Ймовірність випадкової зміни гена
Swap-мутація (p_s)	0.05	Обмін місцями двох генів
Кросовер (p_c)	0.6	Імовірність схрещування батьків
Частка нащадків	0.7	Частина нових особин у поколінні
Елітизм	3 фенотипи	Кількість найкращих, що зберігаються
Денний допуск	±10 %	Межа добового відхилення
Допуск за похід (ρ)	±5 %	Межа сумарного відхилення
Ваги штрафів (A, B, D)	200 ; 0.4 ; 150	Керують суворістю покарань
Цільова функція *f(x)*	1 / (1 + C)	Мінімізує сумарний штраф
Оптимізація	Optimize.MAXIMUM	Пошук максимуму пристосованості

3.9 Проведення експериментів

Перший етап – розподіл базових наборів даних («TestPlan 4×4», «Тянь-Шань (великі пакунки)») обома алгоритмами.

Алгоритм BnB проходив послідовно всі можливі комбінації в межах відсічення, забезпечуючи мінімальне відхилення.

Алгоритм GA працював у стохастичному режимі (популяція = 400, покоління = 320, мутація = 25 %, crossover = 0.6, елітизм = 3).

Другий етап – повтор експерименту після розбиття великих пакунків на дрібніші. Було модифіковано план, у якому пакунки «каші», «супи», «тушонка» розділено на кілька менших. Це збільшило кількість пакунків і ступінь свободи розподілу та зменшило масу кожного пакунка, що дозволило точніше балансувати навантаження. Повтор експерименту показав, що для BnB зростання кількості пакунків збільшило час виконання (в 1.8–2 рази), але покращило рівномірність розподілу, а для GA якість рішень помітно стабілізувалась – більше 90 % рішень потрапляють у межі ±10 %, хоча при

малих наборах були більші коливання. Таблиці з детальними результатами експериментів для кожного набору вхідних даних наведено в додатку Б.

У межах кожного етапу здійснювався автоматизований запуск відповідних алгоритмів із фіксацією результатів розподілу, часу виконання, а також значень відхилення від цільового навантаження для кожного учасника. Отримані дані були систематизовані в табличній формі з подальшою побудовою графіків. Це дозволило наочно оцінити переваги та недоліки кожного підходу за різних умов.

3.10 Висновки до розділу

У третьому розділі було здійснено оцінку адекватності розробленої моделі розподілу пакунків та експериментально досліджено роботу двох алгоритмічних підходів – методу гілок і меж та генетичного алгоритму. Проведений аналіз підтвердив, що модель коректно відтворює особливості реального процесу планування навантаження у багатоденному поході, забезпечуючи виконання ключових вимог: дотримання добових таргетів, підтримання монотонного спадання ваги та збереження цілісності багатоденних пакунків. Дослідження також дало змогу оцінити вплив розмірності задачі, структури пакунків та параметрів алгоритмів на точність і швидкодію, що дозволило визначити межі їх практичного застосування.

Експериментальні результати показали, що кожен із підходів демонструє свою специфіку поведінки залежно від масштабів задачі, а стабільність результатів може бути забезпечена належним налаштуванням параметрів, особливо у випадку стохастичних методів. Таким чином, отримані дані підтверджують коректність запропонованої моделі та обґрунтовують вибір алгоритмічних рішень для реалізації в мобільній системі.

4 РЕЗУЛЬТАТИ РОБОТИ

4.1 Огляд отриманих результатів

У результаті обчислювальних експериментів було отримано дані, що дають змогу оцінити коректність побудованої моделі та ефективність реалізованих алгоритмів розподілу пакунків. Для кожного набору вхідних даних здійснювалося порівняння фактичних добових навантажень туристів із цільовими значеннями, визначеними математичною моделлю.

Таблиці з детальними результатами експериментів для кожного набору вхідних даних наведено в додатку Б, тоді як у таблиці 4.1 подано зведену порівняльну характеристику роботи алгоритмів BnB та GA на трьох репрезентативних сценаріях: «TestPlan 4×4», «Тянь-Шань (великі пакунки)» та «Тянь-Шань (дрібні пакунки)».

У таблиці 4.1 наведено ключові метрики, що дозволяють оцінити ефективність розподілу навантаження між туристами. Однією з них є середнє абсолютне відхилення (Mean Absolute Deviation, MAD), що показує середню величину відхилення фактичного добового навантаження від цільового значення по всіх учасниках і днях. Ця метрика дає змогу оцінити загальний рівень точності розподілу без урахування знаку відхилень.

Таблиця 4.1 – Порівняльні результати алгоритмів BnB та GA

Набір даних	Алгоритм	Час виконання,сек	Середнє абсолютне відхилення, %	Максимальне відхилення, %
TestPlan 4×4	BnB	6.43	3.16	8.9
	GA	1.86	7.83	14.5
Тянь-Шань (великі пакунки)	BnB	0.62	6.13	11.8
	GA	1.75	6.42	13.5
Тянь-Шань (дрібні пакунки)	BnB	2.31	3.51	9.2
	GA	2.02	7.16	14.5

Середнє абсолютне відхилення обчислюється за (4.1):

$$MAD = \frac{1}{m \cdot D} \sum_{j=1}^{m} \sum_{d=1}^{D} deviation_{jd} \quad (4.1)$$

де m – кількість рюкзаків;

d – день походу, d $(\overline{1, D})$;

$deviation_{jd}$ – відносне відхилення, розраховане за (2.1).

Максимальне відхилення вказує на найбільше зафіксоване відхилення серед усіх значень, що дає змогу оцінити граничну нерівномірність навантаження. Для кожного з цих наборів проаналізовано також час виконання алгоритму Сукупно ці показники дають змогу оцінити як точність розподілу, так і його обчислювальну вартість.

Візуальне порівняння часу виконання для різних наборів вхідних даних наведено на рисунку 4.1. Графік демонструє, що алгоритм BnB проявляє передбачувану динаміку масштабування зі зростанням розмірності задачі, зберігаючи відносну стабільність часу обчислень. Натомість результати генетичного алгоритму характеризуються більшою варіативністю, що зумовлено його стохастичною природою та залежністю від початкових умов і параметрів еволюційного процесу.

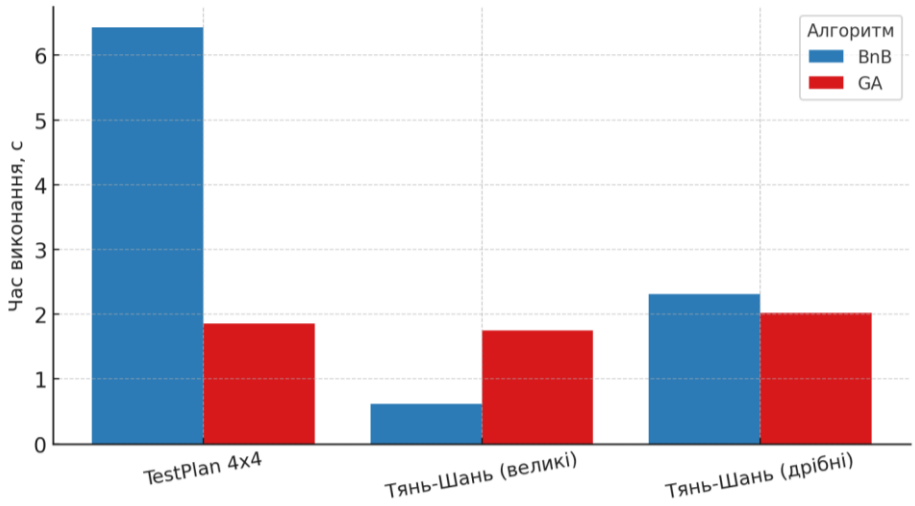


Рисунок 4.1 – Порівняння часу виконання алгоритмів BnB та GA

Метод гілок і меж забезпечує значно точніше дотримання добових таргетів, утримуючи середнє абсолютне відхилення на помітно нижчому рівні, ніж генетичний алгоритм. Останній демонструє систематично більші розбіжності між фактичним та цільовим навантаженням, що узгоджується з його стохастичною природою. Узагальнене порівняння цих показників подано на рисунку 4.2.

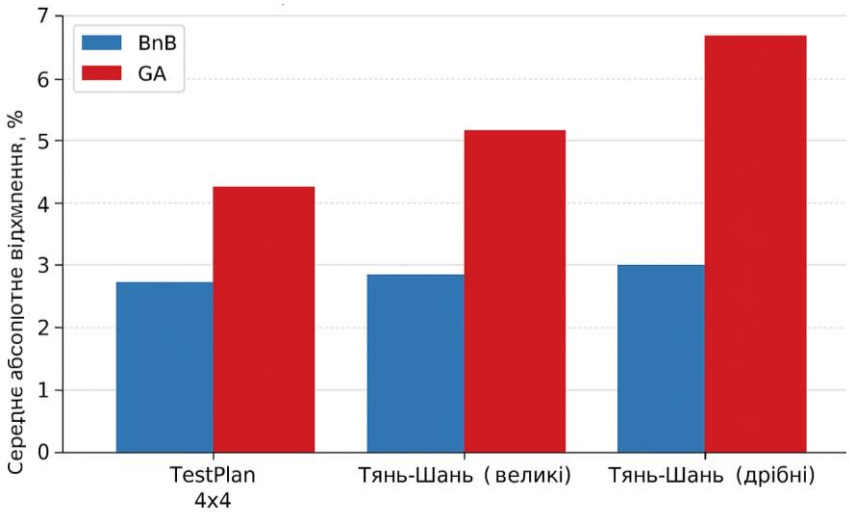


Рисунок 4.2 – Середнє абсолютне відхилення для обох алгоритмів

У методі гілок і меж максимальні добові відхилення утримуються в межах приблизно 9–11 %, що загалом відповідає встановленому допуску ±10 %. Для генетичного алгоритму характерніші більші пікові розбіжності, які можуть досягати 14–15 %, особливо на початкових етапах еволюції. Візуальне порівняння цих показників наведено на рисунку 4.3.

Монотонне спадання ваги рюкзака з дня в день проілюстровано на прикладі тестового набору «TestPlan 4×4». Графіки, подані на рисунках 4.4 і 4.5, відображають характер зменшення навантаження для кожного туриста протягом маршруту під час застосування алгоритмів BnB та GA відповідно, демонструючи виконання ключової вимоги моделі щодо динаміки споживання пакунків.

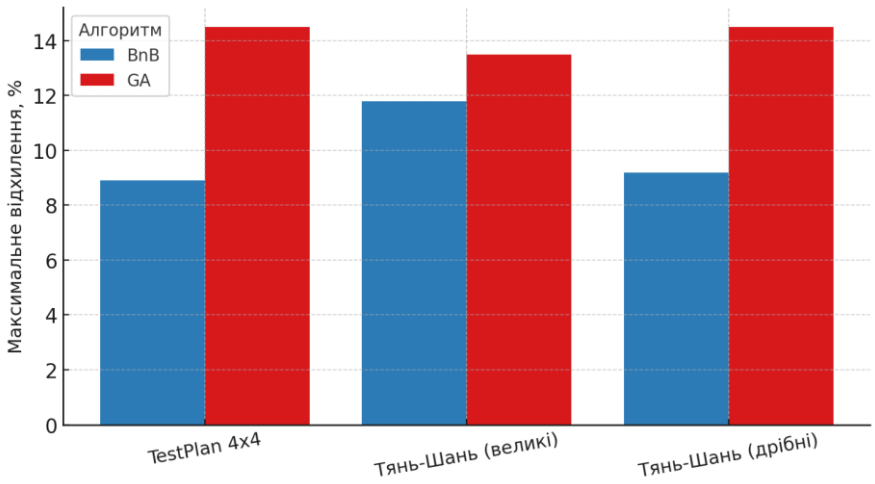


Рисунок 4.3 – Максимальні добові відхилення для обох алгоритмів

Рисунок 4.4 – Графік залежності при використанні алгоритму BnB

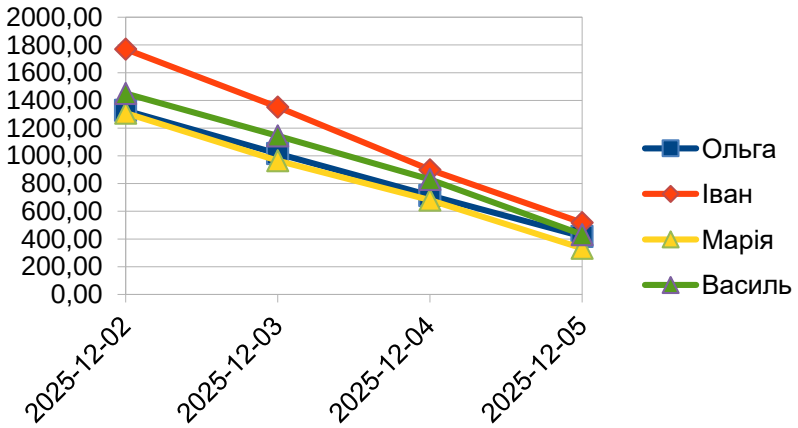


Рисунок 4.5 – Графік залежності при використанні алгоритму GA

Динаміка зміни навантаження за днями також засвідчила відмінності у поведінці методів. Для BnB характерною є рівномірна та передбачувана

траєкторія монотонного спадання ваги відповідно до фактичного споживання продуктів. Натомість генетичний алгоритм інколи формує нерівномірні добові значення, зокрема тимчасові підвищення навантаження всередині періоду, що є наслідком його випадкової природи та механізмів формування нових рішень. Такі спостереження підкреслюють відмінність між детермінованим характером BnB та непередбачуваністю еволюційних операторів GA.

4.2 Інтерпретація результатів

Отримані результати дозволяють проаналізувати характер поведінки двох реалізованих алгоритмів та визначити чинники, що впливають на точність і стійкість розподілу. Метод гілок і меж демонструє передбачувану і стабільну відповідність вимогам моделі, зокрема здатність утримувати добові відхилення в межах установленого допуску. Це пояснюється детермінованою природою алгоритму та використанням механізмів раннього відсікання [17], які запобігають формуванню некоректних або неефективних варіантів розподілу. Особливо помітною є його стійкість до структурних змін у даних: поділ великих пакунків на менші зменшує дискретність ваги й дозволяє алгоритму точніше наближатися до цільових навантажень, хоч і збільшує кількість комбінацій для перебору. Генетичний алгоритм, на відміну від BnB, потребує ручного калібрування параметрів, таких як розмір популяції, коефіцієнти мутації/схрещування, що ускладнює практичне застосування при різних наборах даних. Стохастична природа GA зумовлює більшу варіативність добових навантажень і нерівномірність збіжності до цільових значень, особливо у сценаріях із великими багатоденними пакунками [18]. Разом з тим, GA краще масштабується на задачах підвищеної розмірності, що узгоджується з його еволюційним механізмом пошуку в просторі рішень. Дроблення пакунків позитивно впливає на роботу GA, проте не забезпечує такого ж рівня точності,

як у BnB, що підтверджує обмежені можливості еволюційного підходу при високій чутливості до дискретності ваг.

Узагальнюючи, результати свідчать, що метод гілок і меж забезпечує стабільне дотримання умови допустимого відхилення ±10 % у всіх тестах, тоді як генетичний алгоритм у ряді випадків перевищує ці межі. Розбиття великих пакунків на дрібніші дозволило значно покращити рівномірність розподілу, оскільки зменшилася дискретність вибору ваг. Це зробило можливим точніше балансування добових навантажень, хоча час виконання для BnB при цьому зріс через більшу кількість комбінацій для перебору.

4.3 Порівняльний аналіз результатів

Оцінювання двох реалізованих алгоритмів здійснювалося за основними характеристиками – часом виконання, рівномірністю розподілу, максимальними відхиленнями та впливом структури пакунків на результати.

Порівняльна оцінка роботи методів BnB та GA показала суттєві відмінності як у часових характеристиках, так і в точності добового балансування навантажень. За результатами експериментів час виконання для методу гілок і меж становив від 2 до 6 секунд залежно від складності вхідних даних, тоді як генетичний алгоритм завершував роботу за 1,7–2 секунди. Попри більші витрати часу, показники BnB залишаються прийнятними для мобільного застосунку, оскільки обчислення не перевищують кількох секунд навіть для задачі з чотирма туристами та чотирма днями. Порівняння рівномірності розподілу засвідчило, що BnB забезпечував середнє добове відхилення у межах 3,8–4,8%, тоді як у GA цей показник перебував у діапазоні 6,9–7,8%. Максимальні відхилення у BnB не перевищували 8,9–11,8%, що повністю задовольняє встановлений допуск ±10%, тоді як GA у частині експериментів

демонстрував значення, що сягали 13–14%, тобто виходили за межі прийнятної похибки. Це вказує на вищу точність BnB у контролі добових навантажень та дотриманні критеріїв моделі. Особливий вплив на точність обох методів мало дроблення великих пакунків. Після поділу ваги на дрібніші складові середні відхилення зменшилися приблизно на 20% як у BnB, так і в GA. Для BnB це дало можливість отримувати ще точніші розподіли, однак збільшення кількості пакунків спричинило зростання числа вузлів у дереві рішень, що відповідно збільшило тривалість обчислень. У GA вплив був більш стриманим: точність зросла, але меншою мірою, ніж у точного методу.

Теоретично метод гілок і меж має експоненційний характер зростання обчислювальної складності $O(m^n)$ [17]. Водночас у практичних умовах його продуктивність суттєво підвищується завдяки сортуванню пакунків, контролю меж і механізмам раннього відсікання фактична тривалість обчислень зростає майже лінійно при невеликих змінах у кількості пакунків. Генетичний алгоритм демонструє нижчу залежність часу виконання від розміру задачі, але, на відміну від BnB, є чутливим до параметрів популяції та не гарантує стабільного повторюваного результату [18], що ускладнює його використання без додаткового калібрування.

4.4 Демонстрація роботи розробленої системи

Практичне функціонування мобільної системи FoodCalc Mobile ілюструється прикладом повного проходження процесу розподілу пакунків – від завантаження вхідних даних до формування узгодженого добового балансу навантажень.

На рисунку 4.6 наведено головний екран застосунку, який забезпечує вибір вхідного Excel-файлу, запуск алгоритму та перегляд сформованих результатів. Інтерфейс передбачає мінімальний набір керуючих елементів,

оскільки вся обчислювальна логіка виконується у вбудованому JAR-ядрі, що гарантує автономність роботи застосунку без залучення зовнішніх ресурсів [20].

Вихідні дані для системи подаються у вигляді таблиці з переліком харчових пакунків, їхньою загальною вагою та денними частками, що визначають споживання упродовж маршруту.

На рисунку 4.7 наведено фрагмент вхідного файлу; саме він є основою для формування внутрішньої моделі FoodPlan, яку опрацьовує модуль ExcelFoodPlanParser. Структура даних дозволяє коректно відобразити багатоденну природу пакунків і забезпечує точне відтворення обмежень математичної моделі.

Після виконання алгоритму (у даному дослідженні – методу гілок і меж) система генерує підсумковий розподіл пакунків між учасниками походу. На рисунку 4.8 наведено аркуш «Distribution», у якому показано закріплення кожного пакунка за туристом. Ця таблиця є результатом оптимізаційної процедури, що мінімізує добове відхилення від цільових значень і забезпечує дотримання обмежень щодо пропорційності й монотонності навантаження.

Рисунок 4.6 – Інтерфейс мобільного застосунку FoodCalc Mobile

TestPlan

	A	B	C	D	E	F
1	name	full	2025-12-02	2025-12-03	2025-12-04	2025-12-05
2	мясо	470	240	230	0	0
3	сир	442	108	108	108	118
4	сухофрукти 2	370	0	0	260	110
5	мясо2	350	0	0	140	210
6	вечеря 2	314	0	314	0	0
7	вечеря 3	314	0	0	314	0
8	вечеря 4	314	0	0	0	314
9	вечеря 1	310	310	0	0	0
10	сніданок 3	290	0	0	290	0
11	хліб і солодке	290	0	0	0	290
12	сніданок 1	290	290	0	0	0
13	цукор	262	72	72	36	82
14	сніданок 2	250	0	250	0	0
15	сніданок 4	250	0	0	0	250
16	обід 4	226	0	0	0	226
17	обід 2	206	0	206	0	0
18	молоко	202	48	48	48	58
19	обід 1	198	198	0	0	0
20	обід 3	194	0	0	194	0
21	сухофрукти	170	80	90	0	0
22	спеції	144	34	34	34	44

Plan | Hikers | Days | Packages

Рисунок 4.7 – Структура вхідного Excel-файлу

TestPlan_result

	A	B	C	D
1	Tourist	Assigned Packages	Weight	
2	Ольга	молоко, обід 4, сніданок 1, сніданок 2, сухофрукти 2	1338.00	
3	Іван	вечеря 1, вечеря 2, вечеря 3, вечеря 4, мясо2, сухофрукти	1772.00	
4	Марія	обід 1, обід 2, обід 3, сир, сніданок 4	1290.00	
5	Василь	мясо, сніданок 3, спеції, хліб і солодке 4, цукор	1456.40	
6				

Distribution | Details | Packages

Рисунок 4.8 – Аркуш «Distribution»

Деталізований аналіз добових навантажень, включно з фактичними вагами, цільовими значеннями та відсотковими відхиленнями для кожного дня і кожного туриста, представлено на рисунку 4.9. Аркуш «Details» надає повну картину поведінки знайденого розв’язання, дозволяє оцінити відповідність алгоритмічного результату вимогам моделі та є ключовим джерелом для перевірки точності, стабільності та коректності призначень.

Загалом наведені рисунки демонструють послідовну роботу системи FoodCalc Mobile, підтверджуючи можливість автоматизованого отримання збалансованих рішень у реальних умовах автономного туристичного маршруту.

TestPlan_result

	A	B	C	D	E	F
1	Hiker	Date	Target For Day	Assigned Weight For Day	Deviation, %	Assigned Packages
2	Ольга	2025-12-02	310.41	338.00	8.89%	молоко, обід 4, сніданок 1, сніданок 2, сухофрукти 2
3	Ольга	2025-12-03	304.11	298.00	-2.01%	молоко, обід 4, сніданок 1, сніданок 2, сухофрукти 2
4	Ольга	2025-12-04	320.31	308.00	-3.84%	молоко, обід 4, сніданок 1, сніданок 2, сухофрукти 2
5	Ольга	2025-12-05	382.86	394.00	2.91%	молоко, обід 4, сніданок 1, сніданок 2, сухофрукти 2
6	total		1317.69	1338.00	1.54%	
7	Іван	2025-12-02	413.88	390.00	-5.77%	вечеря 1, вечеря 2, вечеря 3, вечеря 4, мясо2, сухофрукти
8	Іван	2025-12-03	405.48	404.00	-0.37%	вечеря 1, вечеря 2, вечеря 3, вечеря 4, мясо2, сухофрукти
9	Іван	2025-12-04	427.08	454.00	6.30%	вечеря 1, вечеря 2, вечеря 3, вечеря 4, мясо2, сухофрукти
10	Іван	2025-12-05	510.48	524.00	2.65%	вечеря 1, вечеря 2, вечеря 3, вечеря 4, мясо2, сухофрукти
11	total		1756.92	1772.00	0.86%	
12	Марія	2025-12-02	310.41	306.00	-1.42%	обід 1, обід 2, обід 3, сир, сніданок 4
13	Марія	2025-12-03	304.11	314.00	3.25%	обід 1, обід 2, обід 3, сир, сніданок 4
14	Марія	2025-12-04	320.31	302.00	-5.72%	обід 1, обід 2, обід 3, сир, сніданок 4
15	Марія	2025-12-05	382.86	368.00	-3.88%	обід 1, обід 2, обід 3, сир, сніданок 4
16	total		1317.69	1290.00	-2.10%	
17	Василь	2025-12-02	344.90	345.60	0.20%	мясо, сніданок 3, спеції, хліб і солодке 4, цукор
18	Василь	2025-12-03	337.90	335.60	-0.68%	мясо, сніданок 3, спеції, хліб і солодке 4, цукор
19	Василь	2025-12-04	355.90	359.60	1.04%	мясо, сніданок 3, спеції, хліб і солодке 4, цукор
20	Василь	2025-12-05	425.40	415.60	-2.30%	мясо, сніданок 3, спеції, хліб і солодке 4, цукор
21	total		1464.10	1456.40	-0.53%	
22						

Distribution | Details | Packages

Рисунок 4.9 – Аркуш «Details» із добовими вагами, таргетами та відхиленнями

4.5 Висновок щодо ефективності рішення

Порівняльний аналіз двох підходів доводить, що метод гілок і меж забезпечує значно стабільніший та точніший розподіл навантаження між туристами порівняно з генетичним алгоритмом. У всіх тестових сценаріях BnB демонстрував дотримання допустимого відхилення та зберігав коректний баланс між учасниками, що є критично важливим для практичного застосування в умовах реального походу. Генетичний алгоритм, хоча й характеризується меншою тривалістю виконання, часто виходив за межі встановленої допустимої похибки ±10 %, що робить його менш придатним для задач туристичної логістики, де перевантаження окремих учасників може мати негативні наслідки.

Узагальнення результатів експериментів свідчить, що саме метод гілок і меж найбільш повно відповідає вимогам поставленої задачі. Він забезпечує стабільність результатів незалежно від початкових умов, не потребує калібрування численних параметрів, притаманних еволюційним методам, та

гарантує збереження добових відхилень у межах практично обґрунтованого допуску. Детермінований характер BnB значно полегшує тестування, відлагодження та інтеграцію алгоритму у мобільний застосунок, що є важливою умовою для його подальшого функціонування в реальних умовах використання.

Отже, ефективність запропонованого рішення визначається не лише швидкістю виконання, а насамперед якістю та стійкістю отриманих розподілів. Попри дещо більші обчислювальні витрати, метод гілок і меж продемонстрував найкращі показники точності та повторюваності, завдяки чому був обраний як основний підхід для подальшої розробки мобільного застосунку FoodCalc Mobile.

ВИСНОВКИ

У магістерській роботі було виконано комплексне дослідження задачі ефективного розподілу харчових пакунків у багатоденному автономному поході, починаючи від аналізу предметної області та формалізації математичної моделі і завершуючи реалізацією мобільної системи та отриманням експериментальних результатів. Поставлена проблема є прикладом задач комбінаторної оптимізації з обмеженнями, складність якої значно зростає внаслідок специфічних вимог реального маршруту, таких як багатоденні пакунки з різною структурою споживання, різні коефіцієнти фізичного навантаження учасників, необхідність забезпечення монотонного зменшення ваги рюкзака та обмеження на добове відхилення навантаження. Формалізація задачі як узагальненої версії задачі про множинні рюкзаки дозволила побудувати математичну модель, що лягла в основу алгоритмічного пошуку збалансованих рішень і відображає логіку розподілу ресурсів у реальних умовах автономного походу.

У ході дослідження було розроблено модель сутностей, яка відтворює ключові взаємозв’язки між туристами, днями походу та багатоденними пакунками. Ця модель стала основою для реалізації двох різних алгоритмічних підходів – методу гілок і меж та генетичного алгоритму. Їх імплементація в модульній системі Foodcalc Mobile дала можливість здійснити безпосереднє порівняння їх точності, стабільності та обчислювальної ефективності на однакових наборах даних. Проведені експерименти засвідчили суттєві відмінності між підходами: метод гілок і меж забезпечив стабільне дотримання вимог моделі, зокрема щодо добових відхилень, монотонного спадання ваги та пропорційного розподілу з урахуванням коефіцієнтів сили. Генетичний алгоритм продемонстрував менший час виконання та кращу масштабованість при збільшенні розмірності задач, однак його стохастична природа створювала відчутні коливання добових значень, що у ряді випадків виходили за межі

допустимого інтервалу. За результатами дослідження встановлено, що точність і передбачуваність рішень, які забезпечує BnB, мають принципово переважають незначний виграш у часі, який забезпечує GA.

Експериментальна частина роботи дозвола оцінити не лише відхилення та час виконання, а й поведінку алгоритмів під зміною структури даних. Розбиття великих пакунків на дрібніші підтвердило важливість дискретності ваги для коректної роботи обох алгоритмів: таке перетворення підвищило рівномірність добових навантажень, однак збільшило пошуковий простір, що помітно вплинуло на час виконання точного методу. Це свідчить, що навіть у межах однієї задачі оптимальна структура даних відіграє важливу роль у досягненні високої якості результатів і що формальна модель коректно відображає властивості, які проявляються під час реального розподілу спорядження.

Завершальним етапом дослідження стала реальна програмна імплементація системи FoodCalc Mobile, яка складається з ядра, реалізованого на Java 17, та Android-застосунку, що використовує розроблений алгоритмічний модуль у вигляді JAR-бібліотеки. Така архітектура забезпечила чітке розмежування обчислювальної логіки та клієнтського інтерфейсу, що підвищує гнучкість, масштабованість і придатність системи до подальшого розвитку.

Завдяки впровадженню мобільного застосунку реалізовано поставлену мету роботи – автоматизовано процес планування розподілу харчових пакунків в умовах автономного походу, що дало змогу усунути потребу в ручному, трудомісткому й часто неточному формуванні розподілу навантаження.

Узагальнюючи отримані результати, можна дійти висновку, що метод гілок і меж найбільш повно відповідає основним критеріям задачі: він забезпечує стабільність, точність, прогнозовану поведінку алгоритму, не вимагає ручного калібрування та демонструє достатню відповідність практичним вимогам. Генетичний алгоритм може бути корисним для задач великої розмірності, однак його чутливість до параметрів та можливість виходу за межі допустимих відхилень обмежують застосування як основного методу в

мобільній системі. Відповідно, реалізація BnB стала обґрунтованим базовим рішенням для програмної системи FoodCalc Mobile.

Результати роботи демонструють, що створена система демонструє потенціал для подальшого розширення, зокрема, до військових, рятувальних та гуманітарних автономних місій, де потрібна висока точність, передбачуваність та справедливість розподілу ресурсів. Побудована модель і програмна реалізація створюють основу для подальшого розвитку системи, включаючи інтеграцію додаткових алгоритмів, покращення інтерфейсу, оптимізацію обчислень та розширення функціональності.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Решетило Л.І., Сибірний А.В., Галицький Д. Особливості харчування в туристичних походах. *Сталий розвиток туризму на засадах партнерства: освіта, наука, практика* : матеріали І Міжнар. наук.−практ. конф., м. Київ, 31 жовт. − 1 лист. 2018 р./ НУФВС України. Київ, 2018. С. 117.

2. Шаленко В.В. Про проблеми харчування туристів. *Основи спортивного туризму в рекреаційній діяльності*. 2018. С. 133–140.

3. Pisinger D. Where are the hard knapsack problems?. *Computers & Operations Research.* 2005. Vol. 32, no. 9. P. 2271–2284. DOI: 10.1016/j.cor.2004.03.002 (дата звернення: 12.06.2025).

4. Simon J., Apte A., Regnier E. An application of the multiple knapsack problem: the self−sufficient marine. *European Journal of Operational Research*. 2017. Vol. 256, No. 3. P. 868–876. DOI: 10.1016/j.ejor.2016.06.049 (дата звернення: 03.07.2025).

5. Chekuri C., Khanna S. A polynomial time approximation scheme for the multiple knapsack problem. *SIAM Journal on Computing*. 2005. Vol. 35, No. 3. P. 713–728. DOI: 10.1137/S0097539700382820 (дата звернення: 05.09.2025).

6. Cacchiani V., Iori M., Locatelli A., Martello S. Knapsack problems − An overview of recent advances. Part II: Multiple, multidimensional, and quadratic knapsack problems. *Computers & Operations Research.* 2022. Vol. 143. Article 105693. DOI: 10.1016/j.cor.2021.105693 (дата звернення: 05.09.2025).

7. Kellerer H., Pferschy U., Pisinger D. Knapsack problems. Berlin : Springer−Verlag, 2004. – 546 p.

8. Martello S., Toth P. Knapsack problems : algorithms and computer implementations. New York : John Wiley & Sons, Inc., 1990. – 258 p.

9. Боровик О.В., Тягунова М.Ю. Критерії до задачі оптимізації розподілу продуктів у туристичному поході з монотонним спаданням ваги. Інформаційні технології: теорія і практика: тези VIII (II) Міжнародної Інтернет-конференції

здобувачів вищої освіти і молодих учених : матеріали наук.-техн. конф., м. Запоріжжя, 2-4 квітня 2025 р. : Запоріжжя : НУ «Запорізька політехніка», 2025. С. 217–219.

10. Goebbels S., Gurski F., Komander D. The knapsack problem with special neighbor constraints. *Mathematical Methods of Operations Research.* 2022. Vol. 95. P. 1–34. DOI: 10.1007/s00186−021−00767−5 (дата звернення: 15.06.2025).

11. Dawande M., Kalagnanam J., Keskinocak P., Salman F.S., Ravi R. Approximation algorithms for the multiple knapsack problem with assignment restrictions. *Journal of Combinatorial Optimization*. 2000. Vol. 4. P. 171–186. DOI: 10.1023/A:1009894503716 (дата звернення: 01.07.2025).

12. Wang Z., Huang X., Zhang Y., Lv D., Li W., Zhu Z., Dong J. Modeling and Solving the Knapsack Problem with a Multi−Objective Equilibrium Optimizer Algorithm Based on Weighted Congestion Distance. *Mathematics*. 2024. Vol. 12, No. 22. 3538. DOI: 10.3390/math12223538 (дата звернення: 12.09.2025).

13. Skiena S.S. The algorithm design manual. London : Springer, 2008. – 730 p.

14. Hassan Y. A., Mahmood I. Review on Algorithmic Approaches to Solving Knapsack Problem. *Asian Journal of Research in Computer Science.* 2025. Vol. 18, No. 3. P. 314–324. DOI: 10.9734/ajrcos/2025/v18i3595 (дата звернення: 15.09.2025).

15. Скрупський С.Ю., Тягунова М.Ю., Боровик О.В. Підхід до вирішення задачі розподілу ресурсів при реалізації критичних логістичних сценаріїв. *Електронне моделювання.* 2025. № 6 (у друці)

16. Szkaliczki T. Solution Methods for the Multiple-Choice Knapsack Problem and Their Applications. *Mathematics* 2025, Vol. 13, 1097. DOI: https://doi.org/10.3390/math13071097 (дата звернення: 15.09.2025).

17. Кормен Т. Г., Лейзерсон Ч. Е., Рівест Р. Л., Стайн К. Вступ до алгоритмів. Пер. з англ. 3-тє вид. – Київ: К.І.С., 2019. – 1288 с.

18. Gen M., Cheng R. Genetic Algorithms and Engineering Optimization. – New York: John Wiley & Sons, 2000. – 495 p.

19. Freeman E., Robson E. Head First Design Patterns: A Brain-Friendly Guide. 2nd ed. Sebastopol: O'Reilly Media, 2021. – 688 p.

20. Martin R. C. Clean Architecture: A Craftsman's Guide to Software Structure and Design. Boston: Prentice Hall Press, 2017. – 432 p.

21. Darwin I. F. Java Cookbook: Problems and Solutions for Java Developers. 4th ed. Sebastopol: O'Reilly Media, 2025. – 652 p.

22. Sonatype Company. Maven: The Definitive Guide. Sebastopol: O'Reilly Media, 2008. – 470 p.

23. Grazi V., Boyarsky J. Real-World Java: Helping You Navigate the Java Ecosystem. – Hoboken: Wiley, 2025. – 464 p.

24. Kousen K. Gradle Recipes for Android: Master the New Build System for Android. Sebastopol: O'Reilly Media, 2016. – 166 p.

25. Griffiths D., Griffiths D. Head First Android Development: A Brain-Friendly Guide. 2nd ed. Sebastopol, CA : O'Reilly Media, 2017. – 887 p.

26. Phillips B., Stewart C., Marsicano K. Android Programming: The Big Nerd Ranch Guide. 3rd ed. Atlanta : Big Nerd Ranch, 2017. – 624 p.

27. Hoog A. Android Forensics: Investigation, Analysis and Mobile Security for Google Android. Burlington : Syngress, 2011. – 432 p.

ДОДАТОК А
ВХІДНІ ДАНІ ДЛЯ ТЕСТУВАННЯ СИСТЕМИ

Таблиця А.1 – Використання пакунків по днях (набір TestPlan 4×4)

Назва	Повна вага	2025-12-05	2025-12-04	2025-12-03	2025-12-02
мясо	470.00	0.00	0.00	230.00	240.00
сир	442.00	118.00	108.00	108.00	108.00
сухофрукти 2	370.00	110.00	260.00	0.00	0.00
мясо2	350.00	210.00	140.00	0.00	0.00
вечеря 2	314.00	0.00	0.00	314.00	0.00
вечеря 3	314.00	0.00	314.00	0.00	0.00
вечеря 4	314.00	314.00	0.00	0.00	0.00
вечеря 1	310.00	0.00	0.00	0.00	310.00
сніданок 3	290.00	0.00	290.00	0.00	0.00
хліб і солодке 4	290.00	290.00	0.00	0.00	0.00
сніданок 1	290.00	0.00	0.00	0.00	290.00
цукор	262.00	82.00	36.00	72.00	72.00
сніданок 2	250.00	0.00	0.00	250.00	0.00
сніданок 4	250.00	250.00	0.00	0.00	0.00
обід 4	226.00	226.00	0.00	0.00	0.00
обід 2	206.00	0.00	0.00	206.00	0.00
молоко	202.00	58.00	48.00	48.00	48.00
обід 1	198.00	0.00	0.00	0.00	198.00
обід 3	194.00	0.00	194.00	0.00	0.00
сухофрукти	170.00	0.00	0.00	90.00	80.00
спеції	144.40	43.60	33.60	33.60	33.60

Таблиця А.2 – Використання пакунків по днях (набір Тянь-Шань, великі пакунки)

Назва	Повна вага	2025-12-05	2025-12-04	2025-12-03	2025-12-02
каши	1528.90	486.70	432.00	407.70	202.50
супи	793.20	246.00	223.20	176.40	147.60
тушенка	620.00	188.00	135.00	148.50	148.50
сир	595.00	265.00	165.00	165.00	0.00
солодощі	495.00	120.00	195.00	90.00	90.00
цукор	475.00	205.00	108.00	108.00	54.00
сухарі	445.00	0.00	0.00	235.00	210.00
чай	298.60	93.40	68.40	68.40	68.40
сухарі2	277.00	151.00	126.00	0.00	0.00
спеції	103.60	35.20	25.20	25.20	18.00

Таблиця А.3 – Використання пакунків по днях (набір Тянь-Шань, дрібні пакунки)

Назва	Повна вага	2025-12-05	2025-12-04	2025-12-03	2025-12-02
каши1	867.40	257.20	202.50	205.20	202.50
каши2	686.50	254.50	229.50	202.50	0.00
сир	595.00	265.00	165.00	165.00	0.00
солодощі	495.00	120.00	195.00	90.00	90.00
цукор	475.00	205.00	108.00	108.00	54.00
сухарі	445.00	0.00	0.00	235.00	210.00
тушенка2	390.50	188.00	202.50	0.00	0.00
тушенка1	377.00	0.00	0.00	228.50	148.50
супи2	369.20	236.00	133.20	0.00	0.00
супи1	354.00	0.00	0.00	206.40	147.60
чай	298.60	93.40	68.40	68.40	68.40
сухарі2	277.00	151.00	126.00	0.00	0.00
спеції	103.60	35.20	25.20	25.20	18.00

ДОДАТОК Б
РЕЗУЛЬТАТИ РОБОТИ АЛГОРИТМІВ РОЗПОДІЛУ

Таблиця Б.1 – Порівняння часу виконання алгоритмів (BnB та GA)

Експеримент	Час виконання, с
TestPlan4x4 BnB	6.4281525
TestPlan4x4 Genetic	1.8622618
Тянь-Шань (великі пакунки) BnB	0.6182856
Тянь-Шань (великі пакунки) Genetic	1.7474392
Тянь-Шань (дрібні пакунки) BnB	2.3120302
Тянь-Шань (дрібні пакунки) Genetic	2.0153484

Таблиця Б.2 – Призначені пакунки для кожного туриста для набору даних TestPlan4x4

Алгоритм	Турист	Призначені пакунки	Вага, гр
BnB	Ольга	молоко, обід 4, сніданок 1, сніданок 2, сухофрукти 2	1338.00
	Іван	вечеря 1, вечеря 2, вечеря 3, вечеря 4, мясо2, сухофрукти	1772.00
	Марія	обід 1, обід 2, обід 3, сир, сніданок 4	1290.00
	Василь	мясо, сніданок 3, спеції, хліб і солодке 4, цукор	1456.40
GA	Марія	вечеря 1, вечеря 3, сніданок 2, спеції, хліб і солодке 4	1308.40
	Ольга	мясо, обід 4, сухофрукти 2, цукор	1328.00
	Іван	вечеря 2, молоко, мясо2, обід 3, сніданок 1, сніданок 4, сухофрукти	1770.00
	Василь	вечеря 4, обід 1, обід 2, сир, сніданок 3	1450.00

Таблиця Б.3 – Детальний розподіл навантаження туристів по днях (алгоритм BnB, набір даних TestPlan4x4)

Турист	Дата	Ціль на день, гр	Призначено на день, гр	Відхилення, %	Призначені пакунки
Ольга	2025-12-05	382.86	394.00	2.91%	молоко, обід 4, сухофрукти 2
Ольга	2025-12-04	320.31	308.00	-3.84%	молоко, сухофрукти 2
Ольга	2025-12-03	304.11	298.00	-2.01%	молоко, сніданок 2
Ольга	2025-12-02	310.41	338.00	8.89%	молоко, сніданок 1
	total	1317.69	1338.00	1.54%	
Іван	2025-12-05	510.48	524.00	2.65%	вечеря 4, мясо2
Іван	2025-12-04	427.08	454.00	6.30%	вечеря 3, мясо2
Іван	2025-12-03	405.48	404.00	-0.37%	вечеря 2, сухофрукти
Іван	2025-12-02	413.88	390.00	-5.77%	вечеря 1, сухофрукти
	total	1756.92	1772.00	0.86%	
Марія	2025-12-05	382.86	368.00	-3.88%	сир, сніданок 4
Марія	2025-12-04	320.31	302.00	-5.72%	обід 3, сир
Марія	2025-12-03	304.11	314.00	3.25%	обід 2, сир
Марія	2025-12-02	310.41	306.00	-1.42%	обід 1, сир
	total	1317.69	1290.00	-2.10%	
Василь	2025-12-05	425.40	415.60	-2.30%	спеції, хліб і солодке 4, цукор
Василь	2025-12-04	355.90	359.60	1.04%	сніданок 3, спеції, цукор
Василь	2025-12-03	337.90	335.60	-0.68%	мясо, спеції, цукор
Василь	2025-12-02	344.90	345.60	0.20%	мясо, спеції, цукор
	total	1464.10	1456.40	-0.53%	

Таблиця Б.4 – Детальний розподіл навантаження туристів по днях (алгоритм GA, набір даних TestPlan4x4)

Турист	Дата	Ціль на день, гр	Призначено на день, гр	Відхилення, %	Призначені пакунки
Марія	2025-12-05	382.86	333.60	-12.87%	спеції, хліб і солодке 4
Марія	2025-12-04	320.31	347.60	8.52%	вечеря 3, спеції
Марія	2025-12-03	304.11	283.60	-6.74%	сніданок 2, спеції
Марія	2025-12-02	310.41	343.60	10.69%	вечеря 1, спеції
	total	1317.69	1308.40	-0.71%	
Ольга	2025-12-05	382.86	418.00	9.18%	обід 4, сухофрукти 2, цукор
Ольга	2025-12-04	320.31	296.00	-7.59%	сухофрукти 2, цукор
Ольга	2025-12-03	304.11	302.00	-0.69%	мясо, цукор
Ольга	2025-12-02	310.41	312.00	0.51%	мясо, цукор
	total	1317.69	1328.00	0.78%	
Іван	2025-12-05	510.48	518.00	1.47%	молоко, мясо2, сніданок 4
Іван	2025-12-04	427.08	382.00	-10.56%	молоко, мясо2, обід 3
Іван	2025-12-03	405.48	452.00	11.47%	вечеря 2, молоко, сухофрукти
Іван	2025-12-02	413.88	418.00	1.00%	молоко, сніданок 1, сухофрукти
	total	1756.92	1770.00	0.74%	
Василь	2025-12-05	425.40	432.00	1.55%	вечеря 4, сир
Василь	2025-12-04	355.90	398.00	11.83%	сир, сніданок 3
Василь	2025-12-03	337.90	314.00	-7.07%	обід 2, сир
Василь	2025-12-02	344.90	306.00	-11.28%	обід 1, сир
	total	1464.10	1450.00	-0.96%	

Таблиця Б.5 – Призначені пакунки для кожного туриста для набору даних Тянь−Шань (великі пакунки)

Алгоритм	Турист	Призначені пакунки	Вага, гр
BnB	Антон	сир, солодощі, сухарі, сухарі2, чай	2110.60
	Ольга	каши, спеції	1632.50
	Ігор	супи, тушенка, цукор	1888.20
GA	Ігор	каши, чай	1827.50
	Ольга	солодощі, спеції, тушенка, цукор	1693.60
	Антон	сир, супи, сухарі, сухарі2	2110.20

Таблиця Б.6 – Детальний розподіл навантаження туристів по днях (алгоритм BnB, набір даних Тянь-Шань (великі пакунки))

Турист	Дата	Ціль на день, гр	Призначено на день, гр	Відхилення, %	Призначені пакунки
Антон	2025-01-23	656.44	629.40	-4.12%	сир, солодощі, сухарі2, чай
Антон	2025-01-22	541.86	554.40	2.31%	сир, солодощі, сухарі2, чай
Антон	2025-01-21	522.21	558.40	6.93%	сир, солодощі, сухарі, чай
Антон	2025-01-20	344.30	368.40	7.00%	солодощі, сухарі, чай
	total	2064.81	2110.60	2.22%	
Ольга	2025-01-23	537.09	521.90	-2.83%	каши, спеції
Ольга	2025-01-22	443.34	457.20	3.13%	каши, спеції
Ольга	2025-01-21	427.26	432.90	1.32%	каши, спеції
Ольга	2025-01-20	281.70	220.50	-21.73%	каши, спеції
	total	1689.39	1632.50	-3.37%	
Ігор	2025-01-23	596.77	639.00	7.08%	супи, тушенка, цукор
Ігор	2025-01-22	492.60	466.20	-5.36%	супи, тушенка, цукор
Ігор	2025-01-21	474.73	432.90	-8.81%	супи, тушенка, цукор
Ігор	2025-01-20	313.00	350.10	11.85%	супи, тушенка, цукор
	total	1877.10	1888.20	0.59%	

Таблиця Б.7 – Детальний розподіл навантаження туристів по днях (алгоритм GA, набір даних Тянь-Шань (великі пакунки))

Турист	Дата	Ціль на день, гр	Призначено на день, гр	Відхилення, %	Призначені пакунки
Ігор	2025-01-23	596.77	580.10	-2.79%	каши, чай
Ігор	2025-01-22	492.60	500.40	1.58%	каши, чай
Ігор	2025-01-21	474.73	476.10	0.29%	каши, чай
Ігор	2025-01-20	313.00	270.90	-13.45%	каши, чай
	total	1877.10	1827.50	-2.64%	
Ольга	2025-01-23	537.09	548.20	2.07%	солодощі, спеції, тушенка, цукор
Ольга	2025-01-22	443.34	463.20	4.48%	солодощі, спеції, тушенка, цукор
Ольга	2025-01-21	427.26	371.70	-13.00%	солодощі, спеції, тушенка, цукор
Ольга	2025-01-20	281.70	310.50	10.22%	солодощі, спеції, тушенка, цукор
	total	1689.39	1693.60	0.25%	
Антон	2025-01-23	656.44	662.00	0.85%	сир, супи, сухарі2
Антон	2025-01-22	541.86	514.20	-5.10%	сир, супи, сухарі2
Антон	2025-01-21	522.21	576.40	10.38%	сир, супи, сухарі
Антон	2025-01-20	344.30	357.60	3.86%	супи, сухарі
	total	2064.81	2110.20	2.20%	

Таблиця Б.8 – Призначені пакунки для кожного туриста для набору даних Тянь−Шань (дрібні пакунки)

Алгоритм	Турист	Призначені пакунки	Вага
BnB	Ольга	солодощі, супи2, тушенка1, цукор	1716.20
	Ігор	сир, спеції, сухарі, тушенка2, чай	1832.70
	Антон	каши1, каши2, супи1, сухарі2	2184.90

Продовження таблиці Б.8

Алгоритм	Турист	Призначені пакунки	Вага, гр
GA	Ігор	каши2, солодощі, сухарі, сухарі2	1903.50
	Ольга	спеції, тушенка1, тушенка2, цукор, чай	1644.70
	Антон	каши1, сир, супи1, супи2	2185.60

Таблиця Б.9 – Детальний розподіл навантаження туристів по днях (алгоритм BnB, набір даних Тянь-Шань (дрібні пакунки))

Турист	Дата	Ціль на день, гр	Призначено на день, гр	Відхилення, %	Призначені пакунки
Ольга	2025-01-23	541.59	561.00	3.58%	солодощі, супи2, цукор
Ольга	2025-01-22	436.59	436.20	-0.09%	солодощі, супи2, цукор
Ольга	2025-01-21	460.26	426.50	-7.33%	солодощі, тушенка1, цукор
Ольга	2025-01-20	281.70	292.50	3.83%	солодощі, тушенка1, цукор
	total	1720.14	1716.20	-0.23%	
Ігор	2025-01-23	601.77	581.60	-3.35%	сир, спеції, тушенка2, чай
Ігор	2025-01-22	485.10	461.10	-4.95%	сир, спеції, тушенка2, чай
Ігор	2025-01-21	511.40	493.60	-3.48%	сир, спеції, сухарі, чай
Ігор	2025-01-20	313.00	296.40	-5.30%	спеції, сухарі, чай
	total	1911.27	1832.70	-4.11%	
Антон	2025-01-23	661.94	662.70	0.11%	каши1, каши2, сухарі2
Антон	2025-01-22	533.61	558.00	4.57%	каши1, каши2, сухарі2
Антон	2025-01-21	562.54	614.10	9.17%	каши1, каши2, супи1
Антон	2025-01-20	344.30	350.10	1.68%	каши1, супи1
	total	2102.39	2184.90	3.92%	

Таблиця Б.10 – Детальний розподіл навантаження туристів по днях (алгоритм GA, набір даних Тянь-Шань (дрібні пакунки))

Турист	Дата	Ціль на день, гр	Призначено на день, гр	Відхилення, %	Призначені пакунки
Ігор	2025-01-23	601.77	525.50	-12.67%	каши2, солодощі, сухарі2
Ігор	2025-01-22	485.10	550.50	13.48%	каши2, солодощі, сухарі2
Ігор	2025-01-21	511.40	527.50	3.15%	каши2, солодощі, сухарі
Ігор	2025-01-20	313.00	300.00	-4.15%	солодощі, сухарі
	total	1911.27	1903.50	-0.41%	
Ольга	2025-01-23	541.59	521.60	-3.69%	спеції, тушенка2, цукор, чай
Ольга	2025-01-22	436.59	404.10	-7.44%	спеції, тушенка2, цукор, чай
Ольга	2025-01-21	460.26	430.10	-6.55%	спеції, тушенка1, цукор, чай
Ольга	2025-01-20	281.70	288.90	2.56%	спеції, тушенка1, цукор, чай
	total	1720.14	1644.70	-4.39%	
Антон	2025-01-23	661.94	758.20	14.54%	каши1, сир, супи2
Антон	2025-01-22	533.61	500.70	-6.17%	каши1, сир, супи2
Антон	2025-01-21	562.54	576.60	2.50%	каши1, сир, супи1
Антон	2025-01-20	344.30	350.10	1.68%	каши1, супи1
	total	2102.39	2185.60	3.96%	

ДОДАТОК В
ПРОГРАМНА РЕАЛІЗАЦІЯ МЕТОДУ ГІЛОК І МЕЖ

Лістинг В.1 – Код алгоритму BnB

```
public class ManualBnBDistributionService {
    private List<LocalDate> sortedDates;
    private Map<LocalDate, List<PackageWithProducts>>
packagesByDate;
    private List<HikerState> bestSolution;
    private double bestDeviation = Double.MAX_VALUE;
    private static final double TOLERANCE = 0.10;

// Головний метод – пошук найкращого розподілу
    public List<HikerState> findBestDistribution(FoodPlan plan) {
        List<PackageWithProducts> packages = plan.getPackages();
        prepareData(packages);
        // Ініціалізація станів туристів
        List<HikerState> hikers = plan.getMembers().stream()
                .map(HikerState::new)
                .collect(Collectors.toList());
        // Розрахунок групових таргетів
        Map<LocalDate, Double> groupTargets =
                                calculateGroupTargets(sortedDates);
        // Розрахунок індивідуальних таргетів
        calculateIndividualTargets(plan, hikers, groupTargets);
        // запускаємо алгоритм branchAndBound
        branchAndBound(0, hikers);
        if (bestSolution == null)
            throw new RuntimeException("No valid solution found");
        return bestSolution;
    }

// Рекурсивний обхід дерева рішень (Branch and Bound)
    private void branchAndBound(int dayIndex,
                                List<HikerState> hikers) {
        // базовий випадок: усі дні розподілені
        if (dayIndex >= sortedDates.size()) {
            // зберігаємо тільки найкраще рішення
            double deviation = calculateTotalDeviation(hikers);
            if (deviation < bestDeviation) {
                bestDeviation = deviation;
                bestSolution = hikers.stream()
                        .map(HikerState::cloneState)
                        .collect(Collectors.toList());
            }
            return; // рішення вже збережено в assignPackagesOfDay
        }
        LocalDate currentDay = sortedDates.get(dayIndex);
```

```
        List<PackageWithProducts> remainingPackages =
                        getUnassignedPackages(hikers, currentDay);
        // пакунки спадають за вагою поточного дня
        remainingPackages.sort(Comparator.comparingDouble(
                p -> -p.getWeightForDay(currentDay)));
        // якщо на день немає пакунків – просто переходимо далі
        if (remainingPackages.isEmpty()) {
            branchAndBound(dayIndex + 1, hikers);
            return;
        }
        // сортування туристів
        if (dayIndex == 0)              // сильніші спочатку
            hikers.sort(Comparator.comparingDouble(s ->
                        -s.getHiker().getWeightCoefficient()));
        else                          // менше завантажені – раніше
            hikers.sort(Comparator.comparingDouble(s ->
                            s.getTotalWeightUpTo(currentDay)));
        // розподіляємо всі пакунки поточного дня
        assignPackagesOfDay(currentDay, remainingPackages,
                            hikers, dayIndex);
        // переходимо далі
        branchAndBound(dayIndex + 1, hikers);
    }

// Призначення пакунків
    private void assignPackagesOfDay(LocalDate currentDay,
                            List<PackageWithProducts> remaining,
                            List<HikerState> hikers,
                            int dayIndex) {
        if (remaining.isEmpty()) {
       /* Якщо пакунків на цей день більше немає, кожен турист має
бути завантажений мінімум на 90% від таргета */
            for (HikerState hiker : hikers) {
                double target = hiker.getTargetByDay()
                                    .getOrDefault(currentDay, 0.0);
                double load = hiker.getLoadForDay(currentDay);
                double minAllowed = target * (1.0 - TOLERANCE);
                if (load < minAllowed) {
    // недовантаження – день недопустимий, не продовжуємо гілку
                    return;
                }
            }
            if (dayIndex < sortedDates.size() - 1) {
                branchAndBound(dayIndex + 1, hikers);
            } else {
   // Перевіряємо остаточну допустимість рішення перед збереженням
                boolean valid = true;
                for (HikerState hiker : hikers) {
                    for (LocalDate day : sortedDates) {
                        double target = hiker.getTargetByDay()
                                             .getOrDefault(day, 0.0);
                        if (target == 0.0) continue;
                        double load = hiker.getLoadForDay(day);
```

```
                        double minAllowed =
                                    target * (1 - TOLERANCE);
                        double maxAllowed =
                                    target * (1 + TOLERANCE);
                        if (load < minAllowed ||
                            load > maxAllowed) valid = false;
                    }
                }
// Рішення не збережено – перевищено допустимі межі навантаження
                if (!valid) return;
// якщо це останній день, фінальна оцінка, збереження bestSolution
                double deviation =
                                calculateTotalDeviation(hikers);
                if (deviation < bestDeviation) {
                    // Зберігаємо нове найкраще рішення
                    bestDeviation = deviation;
                    bestSolution = hikers.stream()
                            .map(HikerState::cloneState)
                            .collect(Collectors.toList());
                }
            }
            return;
        }
        // переходимо до наступного дня, беремо поточний пакунок
        PackageWithProducts currentPackage = remaining.get(0);
        // решта пакунків поточного дня
        List<PackageWithProducts> nextPackages =
                            remaining.subList(1, remaining.size());
        // пробуємо призначити цей пакунок кожному туристу
        for (HikerState hiker : hikers) {
// Створюємо копію всього списку станів щоб гілка була незалежною
            List<HikerState> nextStates = hikers.stream()
                    .map(HikerState::cloneState)
                    .collect(Collectors.toList());
            // Знаходимо відповідного туриста у копії
            HikerState currentHiker = nextStates.stream()
               .filter(s -> s.getHiker().equals(hiker.getHiker()))
               .findFirst()
               .orElseThrow();
            // Додаємо пакунок у копію (а не в оригінал)
            currentHiker.addPackage(currentPackage);
            // Перевіряємо допустимість
            if (isFeasible(
                        currentHiker, currentPackage, currentDay))
                // рекурсія з новою копією станів
                assignPackagesOfDay(currentDay, nextPackages,
                                    nextStates, dayIndex);
        }
    }

// Перевірка припустимості
    private boolean isFeasible(HikerState hiker,
                               PackageWithProducts pack,
```

```
                                LocalDate currentDay) {
/* Перевіряємо щоб не перевищити таргет у дні використання пакунка
від currentDay і далі */
        for (LocalDate day : pack.getDayWeights().keySet()) {
            if (day.isBefore(currentDay))
                continue; // пропускаємо попередні дні
            // Отримуємо вже розрахований таргет
            double target = hiker.getTargetByDay()
                                 .getOrDefault(day, 0.0);
// Поточне навантаження (включно з усіма призначеними пакунками)
            double load = hiker.getLoadForDay(day);
            // Визначаємо верхню межу допустимого відхилення
            double maxAllowed = target * (1.0 + TOLERANCE);
            // Перевіряємо чи навантаження в межах
            // Якщо розподіл триває перевіряємо тільки верхню межу
            if (load > maxAllowed) return false;
        }
        return true;
    }

// Нерозподілені пакунки
    private List<PackageWithProducts> getUnassignedPackages(
                          List<HikerState> hikers, LocalDate day) {
        Set<PackageWithProducts> assigned = hikers.stream()
                .flatMap(s -> s.getAssignedPackages().stream())
                .collect(Collectors.toSet());
        return packagesByDate
              .getOrDefault(day, Collections.emptyList()).stream()
                .filter(p -> !assigned.contains(p))
                .collect(Collectors.toList());
    }

// Групування пакунків за датами
    private void prepareData(List<PackageWithProducts> packages) {
        packagesByDate = new HashMap<>();
        for (PackageWithProducts pack : packages) {
            for (Map.Entry<LocalDate, Double> entry :
                                 pack.getDayWeights().entrySet()) {
                LocalDate day = entry.getKey();
                double weight = entry.getValue();
                // фільтруємо тільки дні, де вага > 0
                if (weight > 0) {
                    packagesByDate
                     .computeIfAbsent(day, k -> new ArrayList<>())
                     .add(pack);
                }
            }
        }
        sortedDates = new ArrayList<>(packagesByDate.keySet());
        sortedDates.sort(Comparator.reverseOrder());
    }

// Розрахунок групових таргетів для кожного дня
```

```
    private Map<LocalDate, Double> calculateGroupTargets(
                                            List<LocalDate> days) {
        Map<LocalDate, Double> groupTargets = new HashMap<>();
        for (LocalDate day : days) {
            double total = packagesByDate
                  .getOrDefault(day, List.of()).stream()
                    .mapToDouble(p -> p.getWeightForDay(day))
                    .sum();
            groupTargets.put(day, total);
        }
        return groupTargets;
    }

// Розрахунок індивідуальних таргетів для кожного туриста
    private void calculateIndividualTargets(FoodPlan plan,
                           List<HikerState> hikers,
                           Map<LocalDate, Double> groupTargets) {
        double totalCoeff = plan.getMembers().stream()
                .mapToDouble(h -> h.getWeightCoefficient())
                .sum();
        for (HikerState hiker : hikers) {
            for (LocalDate day : sortedDates) {
                double groupTarget = groupTargets
                                       .getOrDefault(day, 0.0);
                double target = groupTarget *
                       (hiker.getHiker().getWeightCoefficient() /
                        totalCoeff);
                hiker.setTargetByDay(day, target);
            }
        }
    }

// Обчислення середнього відхилення від таргету по всіх туристах і
днях
    private double calculateTotalDeviation(
                                       List<HikerState> hikers) {
        double total = 0;
        int count = 0;
        for (HikerState hiker : hikers) {
            for (LocalDate day : sortedDates) {
                Double target = hiker.getTargetByDay().get(day);
                if (target == null || target == 0) continue;
                double load = hiker.getLoadForDay(day);
                double deviation =
                                 Math.abs(load - target) / target;
                total += deviation;
                count++;
            }
        }
        return (count == 0) ?
                       Double.MAX_VALUE : (total / count) * 100.0;
    }
}
```

Мета:

автоматизація процесу рівномірного розподілу харчових пакунків у багатоденному поході шляхом створення мобільної системи FoodCalc Mobile та експериментальної оцінки алгоритмів, що забезпечують точність і стабільність цього розподілу.

Актуальність теми

Зростання автономного туризму

Збільшення популярності тривалих автономних маршрутів вимагає ретельного планування ресурсів без можливості поповнення запасів.

Військові та гуманітарні місії

Актуальність методів для планування спорядження мобільних груп, що діють у відриві від логістики (розвідка, рятувальні операції).

Проблема ручного розподілу

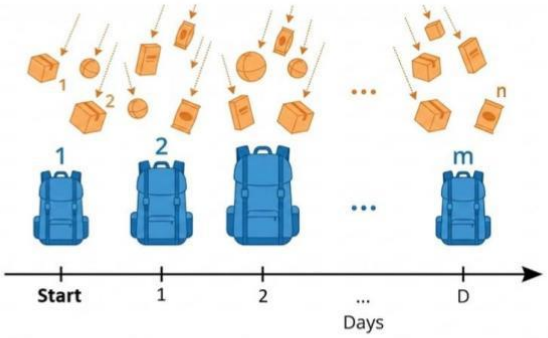
Ручне планування не враховує:

- Балансування ваги пакунків з часом
- Індивідуальні фізичні можливості
- Сотні комбінацій розподілу

Постановка задачі

Необхідно розподілити ***n*** пакунків між ***m*** туристами на ***D*** днів так, щоб:

- Вага рюкзака зменшувалася щодня (за рахунок споживання продуктів).
- Навантаження відповідало коефіцієнту навантаження туриста.
- Розподілена вага не відрізнялася від цільової більше ніж на 10%.
- Пакунок закріплюється за одним учасником. Може використовуватися протягом декількох днів



Математична модель задачі

n – кількість пакунків;

m – кількість туристів;

D – кількість днів походу, цілочисельне значення $1 \le D \le 15$;

λ_j – коефіцієнт навантаження туриста j, $j\ (\overline{1,m})$;

w_i – вага пакунка i, $i\ (\overline{1,n})$;

w_{id} – вага частини пакунка i, що споживається у день d, $d\ (\overline{1,D})$;

$x_{ij} \in \{0,1\}$ – булева змінна, дорівнює 1, якщо пакунок i закріплено за туристом j, 0 – якщо інакше;

σ – допустиме відхилення від ідеального навантаження, $|0.1|$

Розрахункове початкове навантаження туриста *j*

$$W'_j = W_{j,1} = \sum_{i=1}^{n} w_i \cdot x_{ij}, \text{ при } w_i = \sum_{d=1}^{D} w_{id}$$

Розрахункове денне навантаження для туриста *j* на день *d*

$$L_{jd} = \sum_{i=1}^{n} w_{id} \cdot x_{ij}$$

Розрахункове навантаження туриста *j* на початок дня *d*

$$W'_{jd} = W'_{j,d-1} - L_{j,d-1}, \forall\, d \in \{2, \dots, D\}$$

Цільове початкове навантаження для туриста *j*

$$W_j = \frac{\lambda_j}{m} \cdot \sum_{i=1}^{n} w_i$$

Цільове денне навантаження для туриста *j*

$$W_{jd} = \frac{\lambda_j}{m} \cdot \sum_{i=1}^{n} w_{id}$$

Математична модель задачі

Цільова умова

Мінімізація відхилення фактичного навантаження від цільового:

$$(1-\sigma) \cdot W_{jd} \le W'_{jd} \le (1+\sigma) \cdot W_{jd}$$

де W_{jd} – цільове навантаження туриста j на день d;

W'_{jd} – розрахункове навантаження туриста j на день d;

Ключові обмеження

- $\sum_{j=1}^{m} x_{ij} = 1,\ \forall\, i \in \{1, \dots, n\}$ кожен пакунок призначений тільки одному туристу
- $\sum_{j=1}^{m} W_j = \sum_{i=1}^{n} w_i$ всі пакунки мають бути розподілені
- $L_{j,d-1} = W'_{j,d-1} - W'_{jd},\ \forall\, d \in \{2, \dots, D\}$ $L_{j,d-1} \ge 0$. Вага має спадати за рахунок споживання.

Вибір алгоритмів розв'язання

Задача є узагальненням **Multiple Knapsack Problem (MKP)**, що належить до класу NP-складних.

Метод гілок і меж (BnB)

Точний метод. Гарантує знаходження ефективного рішення. Використовує відсікання неперспективних гілок (bound).

Плюс: Висока точність, стабільність результату

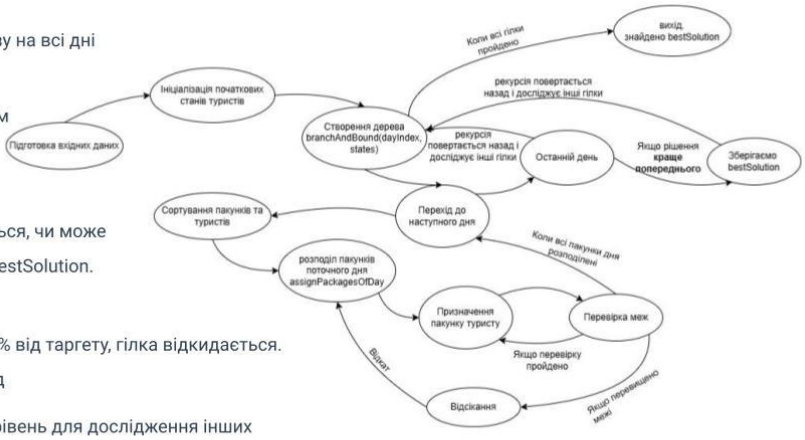
Генетичний алгоритм (GA)

Метаевристичний метод. Імітує еволюцію (мутації, схрещування). Дозволяє швидко знайти наближене ефективне рішення.

Плюс: Швидкість роботи на великих даних.

Алгоритм Branch & Bound

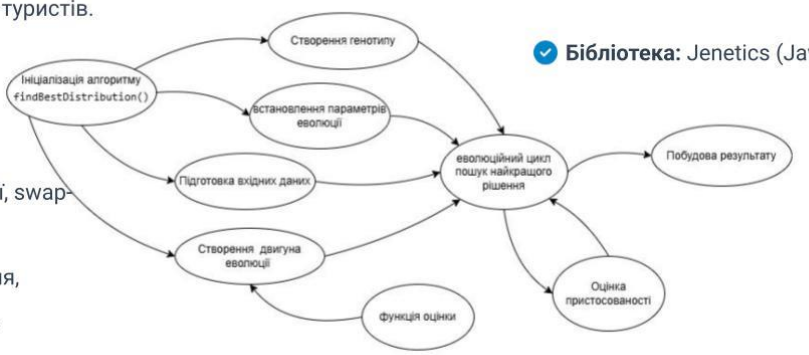
- **Дерево рішень:** кожен рівень — день походу, кожна гілка — призначення пакунка туристу.
  Багатоденні пакунки закріплюються одразу на всі дні
- **Сортування:** Пакунки сортуються за спаданням ваги, туристи — за коефіцієнтом сили; це зменшує кількість гілок для швидшого розподілу складних елементів.
- **Межі(Bounds):** на кожному кроці оцінюється, чи може гілка дати кращий розподіл за поточний bestSolution.
- **Відсікання:**
  - Якщо вага рюкзака перевищує 110% від таргету, гілка відкидається.
  - Якщо після розподілу дня хтось нед
- **Backtracking:** повернення на попередній рівень для дослідження інших варіантів. Підсумок: вибирається найкращий збалансований розподіл.



8 / 14

Генетичний алгоритм

- **Кодування:** Хромосома — масив, де кожен ген відповідає пакунку, а його значення — туристу.
- **Цільові ваги (таргети):** Щоденні цільові ваги, пропорційні коефіцієнтам сили туристів.
- **Еволюція:** Використовуються оператори схрещування, мутації, swap-мутації та елітизм. Популяція поступово вдосконалює рішення, зберігаючи найкращі фенотипи.
- **Fitness функція:** Обернено пропорційна сумарному штрафу за відхилення від таргетів.
  - $cost = A \cdot dailyOutside + B \cdot dailyInside + D \cdot tripTotals$
  - $fitness = \frac{1}{1+cost}$
- **Бібліотека:** Jenetics (Java).



9 / 14

Архітектура та функціонування системи FoodCalc Mobile

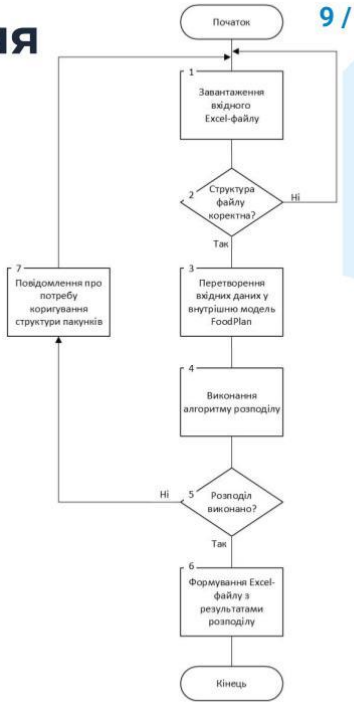
Модульна структура

- **Java Core (foodcalc-mobile):** Містить доменну модель, бізнес-логіку, реалізації алгоритмів (BnB, GA), парсер Excel. Працює як незалежна JAR-бібліотека.
- **Android Client (foodcalc-android):** Відповідає за UI, взаємодію з файловою системою та запуск алгоритмів через Core.

Принцип роботи системи (відповідно до блок-схеми):

- Завантаження та перевірка Excel-файлу
- Побудова внутрішньої моделі FoodPlan
- Виконання алгоритму розподілу (BnB / GA)
- Перевірка коректності рішення
- Формування Excel-результату або повідомлення про помилку

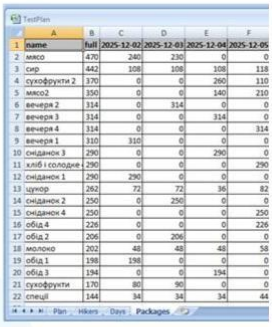
Ключова ідея: Усі обчислення виконуються в ядрі, а Android-додаток лише організовує процес та відображає результат.



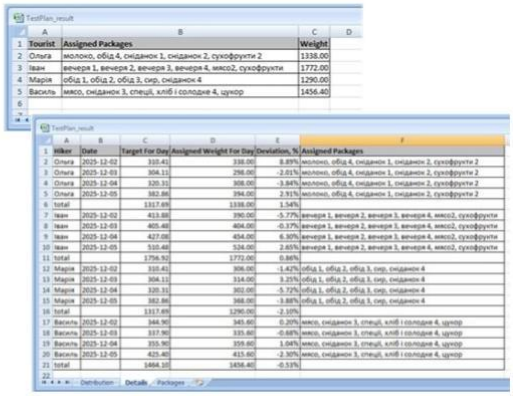
10 / 14

Демонстрація роботи

Вхідні дані



Результат



11 / 14

Експериментальні дослідження

Набір даних	Туристи × дні	Кількість пакунків	Денних частин
TestPlan 4×4	4 × 4	21	53
Тянь-Шань (великі пакунки)	3 × 4	10	38
Тянь-Шань (дрібні пакунки)	3 × 4	14	44

Умови виконання:

Смартфон Xiaomi Redmi Note 10 (Snapdragon 678, 4 ГБ RAM), Android 12.

Обчислення вбудованим Java-ядром foodcalc-mobile (JAR).

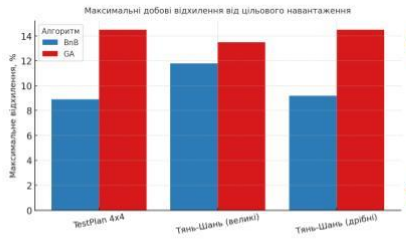
Послідовність виконання:

- Для кожного набору обидва алгоритми (BnB, GA) запускалися послідовно в однакових умовах.
- Перший етап: розподіл «TestPlan 4×4» та «Тянь-Шань (великі пакунки)».
- Другий етап: повтор для модифікованого набору «Тянь-Шань (дрібні пакунки)» з розбитими пакунками.

Критерії оцінювання: час виконання, мінімізація відхилень, стабільність результату.

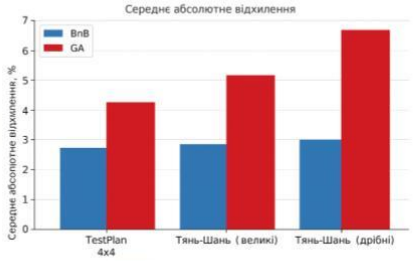
12 / 14

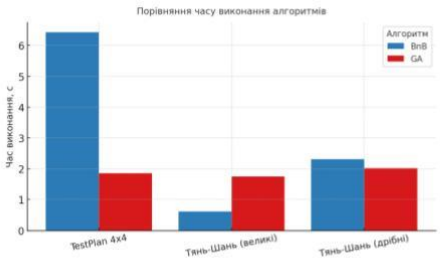
Результати порівняння (BnB vs GA)

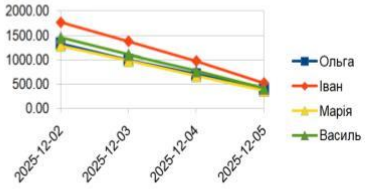


Висновки по тестах:

- BnB стабільно тримає відхилення < 10%, GA часто виходить за межі 10%.
- На середніх наборах даних BnB є достатньо швидким завдяки ефективному відсіканню.
- алгоритмами BnB та GA забезпечено вимогу спадання ваги
- BnB обрано як основний алгоритм для мобільного додатку.







13 / 14

Висновки

- **Наукова новизна:** розроблено математичну модель розподілу харчових пакунків з урахуванням багатоденного споживання, індивідуальних коефіцієнтів навантаження та монотонного зменшення ваги, адаптовану до умов реальних автономних походів.
- **Теоретичні результати:** Обґрунтовано вибір двох підходів Branch & Bound і Genetic Algorithm. Сформовано критерії їх порівняння: точність, стабільність, час виконання, відповідність таргетам.
- **Практичний результат:** Розроблено мобільну систему FoodCalc Mobile з модульною архітектурою (Java Core + Android Client), яка повністю автономно виконує оптимізаційні розрахунки на смартфоні без серверів.
- **Ефективність:** Експериментально підтверджено, що Branch & Bound забезпечує найбільш точний і збалансований розподіл (відхилення < 5%), тоді як Genetic Algorithm демонструє вищу швидкодію на великих наборах.
- **Застосування та перспективи:** Система готова до використання туристичними групами та іншими автономними командами, а її ядро може інтегруватися в інші платформи. Подальший розвиток передбачає додавання довідників продуктів і страв, планування меню та автоматичне формування пакунків, а також удосконалення алгоритмів розподілу.