УДК 004.9: 519.81

Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет *Комп'ютерних наук*
(повна назва)

Кафедра *Системотехніки*
(повна назва)

# АТЕСТАЦІЙНА РОБОТА
# Пояснювальна записка

Рівень вищої освіти *другий (магістерський)*

ГЮИК.503100.003_*ПЗ*

*Методи визначення підмножин ефективних варіантів у системах підтримки прийняття проектних рішень*
(тема)

Виконав:
Студент *2* курсу, групи *СПРм-18-2*
Спеціальність *122 – Комп'ютерні науки*
(код і повна назва напряму)
Тип програми *освітньо-наукова*
(освітньо-професійна або освітньо-наукова)
Освітня програма *Системне проектування*
(повна назва освітньої програми)

*Вакуленко В.К.*
(прізвище, ініціали)

Керівник *Безкоровайний В.В.*
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ *Гребеннік І. В.*
(підпис) (прізвище, ініціали)

2020 р.

*Атестаційна робота оформлена у відповідності до вимог діючих стандартів та методичних вказівок.*

*Матеріали атестаційної роботи не містять відомостей, що заборонені для опублікування у відкритих виданнях.*

*Попередній захист проведено.*

*Керівник атестаційної роботи* *В.В.Безкоровайний*

20.05.20

Харківський національний університет радіоелектроніки

Факультет *Комп'ютерних наук*

(повна назва)

Кафедра *Системотехніки*

(повна назва)

Рівень вищої освіти другий (магістерський)

Спеціальність *122 – Комп'ютерні науки*

(код і повна назва)

Тип програми *Освітньо-наукова*

(освітньо-професійна або освітньо-наукова)

Освітня програма *Системне проектування*

(повна назва)

ЗАТВЕРДЖУЮ:

Зав. кафедри _____

(підпис)

«____»_____ 20___ р.

# ЗАВДАННЯ

НА АТЕСТАЦІЙНУ РОБОТУ

студентові *Вакуленко Вікторії Костянтинівні*

(прізвище, ім'я, по батькові)

1. Тема роботи *«Методи визначення підмножин ефективних варіантів у системах підтримки прийняття проектних рішень»*

затверджена наказом по університету від « *30* » *03* 20*20* р. № *477 Ст*

2. Термін подання студентом роботи (проекту) *26 травня 2020 р.*

3. Вихідні дані до роботи (проекту) *Структура файлу з характеристиками проектних варіантів: кількість часткових критеріїв – до 5; кількість варіантів – до 20000. Технічне забезпечення: ІВМ-сумісний персональний комп'ютер. Вхідні дані – можливість завантаження з пам'яті та генерації. Вихідні данні: час розв'язання задачі; потужність і склад підмножини ефективних варіантів.*

4. Зміст пояснювальної записки (перелік питань, що потрібно розробити) *Вступ. Огляд сучасного стану проблеми визначення підмножин ефективних проектних рішень. Етапи, процедури, методи прийняття проектних рішень. Постановка мети та завдань дослідження. Методи й алгоритми визначення підмножин ефективних варіантів. Розробка програмного засобу. Експерименти та аналіз результатів. Висновки.*

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслеників, плакатів)
*Плакати або кресленики на аркушах формату А4 по тексту записки та/або в додатках: схеми алгоритмів методів визначення підмножин ефективних проектних варіантів; графічне подання часової складності методів і алгоритмів розв'язання задачі.*

6. Консультанти розділів роботи (проекту)

| Найменування розділу | Консультант (посада, прізвище, ім'я, по батькові) | Позначка консультанта про | |
|---|---|---|---|
| | | підпис | дата |
| | | | |
| | | | |

## КАЛЕНДАРНИЙ ПЛАН

| № | Назва етапів роботи | Терміни виконання етапів роботи | Примітка |
|---|---|---|---|
| 1 | *Отримання, аналіз завдання, уточнення* | *30.03.2020* | |
| | *плану роботи* | | |
| 2 | *Аналіз сучасного стану проблеми* | | |
| | *підтримки прийняття проектних рішень* | *03.04.2020* | |
| 3 | *Огляд існуючих методів розв'язання задачі* | *06.04.2020* | |
| 4 | *Розробка математичного забезпечення задачі* | *13.04.2020* | |
| 5 | *Розробка програмного забезпечення задачі* | *20.04.2020* | |
| 6 | *Проведення експериментальних досліджень* | *28.04.2020* | |
| 7 | *Оформлення пояснювальної записки* | *03.05.2020* | |
| 8 | *Підготовка презентації* | *09.05.2020* | |
| 9 | *Подання закінченої роботи науковому* | *14.05.2020* | |
| | *керівникові* | | |
| 10 | *Подання роботи на рецензування* | *21.05.2020* | |
| 11 | *Попередній захист* | *22.05.2020* | |
| 12 | *Подання роботи до екзаменаційної комісії* | *25.05.2020* | |

Дата видачі завдання «*30*» *березня* *2020 р.*

Студент _____ *Вакуленко В.К.*
(підпис)

Керівник роботи _____ *проф. Безкоровайний В.В.*
(підпис) (посада, прізвище, ініціали)

# РЕФЕРАТ

Пояснювальна записка до магістерської атестаційної роботи: 102 с., 9 табл., 15 рис., 4 додатки, 56 джерел інформації.

МЕТОД, МНОЖИНА ЕФЕКТИВНИХ ВАРІАНТІВ, ПРОЕКТНЕ РІШЕННЯ, СИСТЕМА ПІДТРИМКИ ПРИЙНЯТТЯ РІШЕНЬ.

Об’єктом дослідження є системи підтримки прийняття проектних рішень.

Предметом досліджень є ефективність процедур виділення підмножин ефективних варіантів у системах підтримки прийняття проектних рішень.

Метою даної роботи є підвищення ефективності систем підтримки прийняття багатокритеріальних проектних рішень за рахунок удосконалення процедур виділення підмножин ефективних варіантів.

Методи дослідження – у роботі використовується методи попарних порівнянь, виділення наближеної множини компромісів, Карліна, Гермейєра, еволюційний на основі генетичного алгоритму NSGA-II, випадкового пошуку.

У роботі удосконалено еволюційний метод виділення підмножин ефективних проектних рішень, побудований на основі генетичного алгоритму, у частині зменшення його часової складності та підвищення точності. Розроблено алгоритми та програмне забезпечення розв’язання задачі, що дозволило провести експериментальне дослідження розглянутих та удосконаленого методу.

Галузь застосування – підсистеми прийняття рішень за множиною показників якості у компаніях, організаціях та установах, що здійснюють в своїй діяльності вирішення завдань автоматизованого проектування, планування розвитку чи управління.

ABSTRACT

Thesis contains: 102 pages, 9 tables, 15 figures, 4 applications, 56 sources.

DECISION SUPPORT SYSTEM, DESIGN DECISION, METHOD, SET EFFECTIVE OPTIONS.

The object of research is the systems of project decision support.

The subject of research is the effectiveness of procedures for the allocation of subsets of effective options in project decision support systems.

The purpose of this work is to increase the efficiency of support systems for multicriteria design decisions by improving the procedures for allocating subsets of effective options.

Research methods – the work uses methods of pairwise comparisons, selection of an approximate set of compromises, Carlin, Hermeyer, evolutionary on the basis of the genetic algorithm NSGA-II, random search algorithm.

The paper improves the evolutionary method of selecting subsets of effective design solutions, built on the basis of a genetic algorithm, in terms of reducing its time complexity and increasing accuracy. Algorithms and software for solving the problem have been developed, which allowed to conduct an experimental study of the considered and improved method.

Scope – decision-making subsystems for many quality indicators in companies, organizations and institutions that carry out in their activities to solve problems of computer-aided design, development planning or management.

ЗМІСТ

# ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ІУ – індекс узгодженості.

МАІ – метод аналізу ієрархій.

ОПР – особа, яка приймає рішення.

САПР – система автоматизації проектувальних робіт.

СППР – система підтримки прийняття рішень.

# ВСТУП

Прийняття рішень постійно супроводжує процес людської діяльності. Цей процес зачіпає різні галузі, такі як проектування, технології, економіка, менеджмент і багато інших. Процес прийняття рішення – це складний процес, який ґрунтується не тільки на об'єктивних кількісних ознаках, але і на суб'єктивних судженнях експертів, які приймають участь в прийнятті рішень. У сучасних реаліях не завжди існує можливість для експертів прийняти рішення без використання допоміжних засобів, так як складність завдань, кількість вхідних даних сильно зросла за останні десятиліття. У більшості випадків, рішення слід приймати в умовах часових обмежень.

Все це призводить до того, що потрібен додатковий інструментарій, наприклад, системи підтримки прийняття рішень. Функціональність систем підтримки прийняття рішень (СППР) повинна включати інтуїтивно зрозумілі предметно-орієнтовані засоби подання, аналізу та моделювання корпоративних процесів. Ці засоби повинні передбачати налагодження на аналітичний профіль користувача. Методики багатовимірного аналізу повинні бути не тільки зручними, але й стійкими при роботі в інформаційних розрізах з великим історичним горизонтом.

При використанні СППР виникають ситуації, коли кількість ефективних рішень набагато більше ніж експерт або група експертів можуть обробити. Подібні ситуації виникають при наявності множини часткових критеріїв та великої кількості альтернатив.

Метою даної роботи є підвищення ефективності систем підтримки прийняття багатокритеріальних проектних рішень за рахунок удосконалення процедур виділення підмножин ефективних варіантів.

Для досягнення поставленої мети необхідно вирішити такі завдання:

– провести аналіз сучасного стану проблеми підтримки прийняття проектних рішень;

– обрати базові методи виділення підмножин ефективних проектних рішень;

– удосконалити або модифікувати базовий метод виділення підмножин ефективних проектних рішень;

– програмно реалізувати удосконалений (модифікований) метод виділення підмножин ефективних проектних рішень;

– провести експериментальне порівняльне дослідження ефективності удосконаленого (модифікованого) методу виділення підмножин ефективних проектних рішень;

– надати рекомендації щодо практичного використання удосконалених (модифікованих) методів виділення підмножин ефективних проектних рішень у системах підтримки прийняття проектних рішень.

Об’єктом дослідження є системи підтримки прийняття проектних рішень.

Предметом досліджень є ефективність процедур виділення підмножин ефективних варіантів у системах підтримки прийняття проектних рішень.

Методи дослідження – у роботі використовуються методи попарних порівнянь, методи на основі згорток Карліна та Гермейєра, еволюційний метод на основі генетичного алгоритму NSGA-II, метод випадкового пошуку.

У роботі удосконалено еволюційний метод виділення підмножин ефективних проектних рішень, побудований на основі генетичного алгоритму, у частині зменшення його часової складності та підвищення точності. Розроблено алгоритми та програмне забезпечення розв’язання задачі, що дозволило провести експериментальне дослідження розглянутих та удосконаленого методу.

Практичне значення одержаних результатів полягає у наданні можливості скорочення часу реалізації процедур виділення підмножин ефективних варіантів у системах підтримки прийняття проектних рішень.

За результатами дослідження підготовлено доповіді на: 24-й Міжнародний молодіжний форум «Радіоелектроніка і молодь в XXI столітті» (м. Харків); звітну наукову конференцію викладачів, докторантів, аспірантів та студентів Прикарпатського національного університету ім. В. Стефаника (м. Івано-

Франківськ); міжнародну науково-практичну конференцію «Інформаційні технології в культурі, мистецтві, освіті, науці, економіці та бізнесі» (м. Київ).

# 1 ОГЛЯД СУЧАСНОГО СТАНУ ПРОБЛЕМИ ВИЗНАЧЕННЯ ПІДМНОЖИН ЕФЕКТИВНИХ ПРОЕКТНИХ РІШЕНЬ

## 1.1 Процедури прийняття рішень

Проектування будь-якої системи передбачає прийняття ефективних рішень. Прийняття рішень є складною процедурою, методи реалізації і оптимізації якої розглядаються в різних наукових дисциплінах.

Першим етапом процедури прийняття рішень є постановка завдання. Постановка завдання включає формулювання цілей, а також дій і умов, в яких їх необхідно виконати [1].

Наступними етапами прийняття рішення є: розробка моделей задач, вибір і прийняття рішення. На другому етапі визначається характеристика завдань відповідно до класифікації, а також створюються відповідні математичні моделі. На третьому етапі проводиться визначення допустимих рішень на основі заданих обмежень при використанні систем підтримки прийняття рішень. Далі йде визначення оптимальних рішень з використанням розроблених раніше математичних моделей [2].

У загальному вигляді, завдання прийняття рішень (ЗПР) можна подати як формування множини можливих варіантів, які забезпечують розв’язання поставленої задачі при існуючих обмеженнях, і виділення серед цих варіантів одного кращого або декількох кращих варіантів, що задовольняють поставленим вимогам [3].

Формально задачу прийняття рішення $D$ можна записати у такому узагальненому вигляді [3]:

$$D = \langle F, A, X, G, P \rangle, \tag{1.1}$$

де $F$ – формулювання задачі прийняття рішень, яка включає в себе визначення цілей і вимог;

$A$ – сукупність можливих альтернатив, з яких буде проводитися вибір;

$X$ – сукупність критеріїв, що описують альтернативи;

$G$ – сукупність обмежень, які обмежують множину (область) допустимих варіантів;

$P$ – переваги особи або осіб, що приймають рішення, які служать для оцінки та порівняння можливих варіантів рішення.

Проектуванні систем довгий і складний процес, при якому грають важливу роль величезна кількість факторів, які в тій чи іншій мірі будуть впливати на результат. Так як рівень глобалізації з кожним роком зростає, системи стають все більш високонавантаженими, розподіленими, спрямованими на вирішення більш складних завдань. Експертам, задіяним в процесі проектування, слід враховувати множини критеріїв і альтернатив, за якими вони будуть здійснювати прийняття рішень. Подібні завдання називають багатокритеріальними. Такий клас задач є найскладнішим, і в той же час одним з найбільш поширених. Крім проектування вони широко застосовується в інших галузях людської діяльності, таких як економіка, управління, інженерія.

В багатокритеріальних задачах прийняття рішень передбачається, що існує множина альтернатив з декількома атрибутами, які особа або особи, які приймають рішення повинні аналізувати і оцінювати. Метою багатокритерільнного прийняття рішень є пошук найкращої альтернативи або ж ранжування можливих альтернатив для підтримки прийняття рішень.

Класичними методами є адитивний метод зважування, метод аналізу ієрархій і метод упорядкування переваг за подібністю з ідеальним рішенням. У подібних завданнях необхідними вхідними даними для прийняття рішень є значення критеріїв і їх вага. Вага критерію відображає його важливість. Через складність реального світу і обмежених можливостей людини в пізнанні і сприйнятті ці значення часто не можуть бути визначені точно [4].

Так як часто особа, яка приймає рішення (ОПР) не може висловити свої переваги в кількісному вигляді, це приводить до створення пласта завдань, спрямованих на автоматизацію та оптимізацію процесу підрахунку вагових коефіцієнтів критеріїв.

Математичні моделі, що використовуються при прийнятті рішень, стають більш складними, створюються нові моделі або вдосконалюються існуючі.

Прикладом може виступати багатокритеріальна модель, що використовується при проектуванні будинків [5]. Моделі цієї предметної області характеризується великою кількістю критеріїв, обмеженнями, що часто змінюються, а також жорсткими часовими рамками. Множина ефективних рішень отриманих за визначених умов може містити величезну кількість допустимих варіантів. Використання багатокритеріальної моделі допомагає особам, які приймають рішення, вибрати найбільш ефективну альтернативу серед величезної кількості Парето-оптимальних рішень.

У найпростішому випадку рішення щодо вибору найкращої альтернативи за одним критерієм може здійснювати ОПР без використання систем підтримки прийняття рішень. При значній кількості критеріїв оцінювання і альтернатив, стає практично неможливо зробити об'єктивне порівняння і оцінку одним експертом. У більшості випадків при прийнятті рішень бере участь група експертів, кожен з яких має своє власне уявлення про важливість критеріїв, спираючись на попередньому досвіді [6]. У таких ситуаціях слід застосувати математичні моделі, які зможуть акумулювати переваги групи експертів.

Для усереднення групового рішення експертів можна використовувати наступну матрицю [6]:

$$\tilde{V} = \left(\tilde{v}_{ij}\right)_{m\times n} = \begin{matrix} \\ A_1 \\ A_2 \\ \vdots \\ A_m \end{matrix} \begin{matrix} \begin{matrix} u_1 & u_2 & \dots & u_n \end{matrix} \\ \begin{bmatrix} \tilde{v}_{11} & \tilde{v}_{12} & \dots & \tilde{v}_{1n} \\ \tilde{v}_{21} & \tilde{v}_{22} & \dots & \tilde{v}_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ \tilde{v}_{m1} & \tilde{v}_{m2} & \dots & \tilde{v}_{mn} \end{bmatrix} \end{matrix}, \tag{1.2}$$

де $m$ – кількість альтернатив $A_i$;

$n$ – кількість критеріїв $u_j$;

$\widetilde{v_{ij}}$– множина суджень по кожному рішенню, що приймається.

Виходячи з (1.2) наближена матриця групових рішень $RV$ матиме наступний вигляд:

$$RV = \begin{array}{c} \\ A_1 \\ A_2 \\ \vdots \\ A_m \end{array} \begin{array}{c} \begin{array}{cccc} u_1 & u_2 & \dots & u_n \end{array} \\ \begin{bmatrix} [\tilde{v}_{11}{}^L, \tilde{v}_{11}{}^U] & [\tilde{v}_{12}{}^L, \tilde{v}_{12}{}^U] & \dots & [\tilde{v}_{1n}{}^L, \tilde{v}_{1n}{}^U] \\ [\tilde{v}_{21}{}^L, \tilde{v}_{21}{}^U] & [\tilde{v}_{22}{}^L, \tilde{v}_{22}{}^U] & \dots & [\tilde{v}_{2n}{}^L, \tilde{v}_{2n}{}^U] \\ \vdots & \vdots & \vdots & \vdots \\ [\tilde{v}_{m1}{}^L, \tilde{v}_{m1}{}^U] & [\tilde{v}_{m2}{}^L, \tilde{v}_{m2}{}^U] & \dots & [\tilde{v}_{mn}{}^L, \tilde{v}_{mn}{}^U] \end{bmatrix} \end{array}. \quad (1.3)$$

Як випливає з вищеописаного багатокритеріальні задачі мають широкий спектр завдань, який вимагає дослідження. Часто подібні завдання вимагають великих затрат часу, а також в них стикаються з неможливістю виразити критерії в кількісному вигляді і об'єктивно оцінити їх. Ц вимагає, в свою чергу, пошуку нових методів вирішення або удосконалення існуючих. Подібні завдання широко поширені в сучасному світі та знаходять застосування практично у всіх сферах людського життя, таких як технології, управлінське планування, проектування систем, медицина, економіка тощо.

### 1.1.1 Етапи процесу прийняття рішень

В теорії прийняття рішень процес прийняття рішення можна розділити на наступні етапи:

– виявлення проблеми проектування та постановка завдання;

– формування проектних альтернатив і їх наслідків;

– вибір альтернативного проектного рішення;

– аналіз результатів прийнятого проектного рішення.

Ці етапи можуть мати ітеративний характер. Так при отриманні проектного рішення може бути змінено кількість альтернатив.

Відповідно до [14] формальні методи прийняття проектних рішень можуть успішно застосовуватися для таких випадків:

– при вирішенні проблеми або проблемної ситуації результат може ототожнюватися з однією або декількома цілями;

– є кілька альтернатив (варіантів вирішення проектної проблеми, способів досягнення цілей). Альтернатив в задачах прийняття рішень завжди більше однієї;

– є критерії, які накладають обмеження на варіанти вирішення поставленої проектної проблеми або проблемної ситуації;

– існує людина або група осіб, які зацікавлені у вирішенні проектної проблеми, які володіють повноваженнями вибору рішення, а так само несуть відповідальність за його виконання.

У процесі прийняття проектного рішення [15] найчастіше беруть участь керівник проектної групи і група фахівців, які беруть участь на різних етапах прийняття рішення. Для слабо структурованих завдань виділяють власника проектної проблеми (ВП) і особу, яка приймає рішення ОПР. Не завжди власник проблеми є ОПР. У деяких ситуаціях ВП – це частина групи осіб, які приймають рішення. Іноді в процесі вирішення проектної проблеми привертають експертів, які мають більш глибоке уявлення про аспекти даної проблеми.

У процесі прийняття рішень основними є такі поняття як альтернативи і критерії.

Альтернативами [16] називають варіанти застосовуваних рішень, в рамках завдань прийняття рішення їх має бути дві або більше. Їх поділяють на залежні і незалежні. Незалежними називають ті альтернативи, які не впливають на якість інших альтернатив. На відміну від них при залежних альтернативах рішення по одній з них впливає на якість інших.

Критерії прийняття рішень – це спосіб опису альтернатив, спосіб вираження відмінностей між ними з точки зору переваг ОПР. Сучасні методи прийняття рішень орієнтовані на врахування всіх відмітних ознак альтернатив, що наближає формальні схеми до реального світу. Тому багатокритеріальний

опис альтернатив набуває все більшого поширення. Критерії, як правило, задаються не на початковому етапі аналізу проблеми, а в процесі діалогу між ОПР і експертом.

Залежно від впливу критерію на оцінку альтернативи по іншому критерію розрізняють залежні і незалежні критерії оцінки. За аналогією з альтернативами, критерій називається залежним тоді, коли оцінка альтернативи по ньому визначає ймовірнісну оцінку за іншим критерієм. Завдання прийняття рішень і методи їх вирішення залежать від кількості критеріїв.

Виявлення переліку альтернатив та критеріїв є першим і необхідним етапом задач прийняття рішень.

### 1.1.2 Архітектура та класифікація СППР

Узагальнена архітектура системи підтримки рішень [12] складається з 5 частин: система управління даними, система управління моделями, машина знань, інтерфейс користувача та користувачі системи.

Системи підтримки прийняття рішень, як і системи підтримки вибору рішень, забезпечують необхідною інформацією особу, яка приймає рішення, для прийняття індивідуальних і групових рішень на основі комп'ютерного моделювання. При цьому дані можуть надходити з проектуючих підсистем САПР, з власної бази даних, а також зі сховищ даних. Ефективність використання СППР обумовлена, перш за все, можливістю ОПР розглядати значну кількість альтернатив, використовувати моделі при аналізі інформації, формуванні різних альтернатив і їх оцінки за обраними критеріями, а також оцінки наслідків прийнятого рішення.

В основний функціональний набір СППР сьогодні входять [13]:

– фінансове планування і бюджетування;
– формування консолідованої звітності;

– створення інформаційної системи проектування на основі ключових показників діяльності з заздалегідь настроєними бібліотеками показників; аналіз взаємовідносин з розробниками і постачальниками; аналіз ринкових тенденцій;

– функціонально-вартісний аналіз;

– функціонально-вартісне управління;

– система постійних поліпшень;

– багатовимірний аналіз даних;

– виявлення прихованих закономірностей;

– виявлення моделей (структур) даних; статистичний аналіз і прогнозування часових рядів;

– подієве управління процесом проектування;

– аналіз ризиків;

– формування заздалегідь настроєних запитів; інтелектуальний пошук (за неповними даними та неформальним запитам);

– бізнес-моделювання та аналіз ефективності виконання бізнес-процесів; референтні галузеві моделі.

Виділяють три класи СППР [13] у залежності від складності вирішуваних завдань і областей застосування.

СППР першого класу, що володіють найбільшими функціональними можливостями, призначені для застосування в органах, де приймаються рішення державного рівня і органах управління великих компаній (рада директорів корпорації) при плануванні великих комплексних цільових програм для обґрунтування рішень щодо включення в програму різних політичних, соціальних або економічних заходів і розподілу між ними ресурсів на основі оцінки їх впливу на досягнення основної мети програми. СППР цього класу є системами колективного користування, бази знань яких формуються багатьма експертами – фахівцями в різних областях знань.

СППР другого класу є системами індивідуального користування, бази знань яких формуються безпосереднім користувачем. Вони призначені для

використання співробітниками середнього рівня, а також керівниками малих та середніх фірм для вирішення завдань проектування й управління.

СППР третього класу є системами індивідуального користування, що адаптуються до досвіду користувача. Вони призначені для розв’язання задач системного аналізу, проектування й управління. Такі системи забезпечують отримання рішення поточної задачі на основі інформації про результати практичного використання рішень цієї ж задачі, прийнятих в минулому.

### 1.1.3 Розвиток і використання систем підтримки прийняття рішень

Сильною стороною систем підтримки прийняття рішення є те, що вони дозволяють поєднувати теорію та практику. Оскільки СППР описує реально існуючі системи, які використовуються на різних підприємствах, з працюючими там експертами, та теоретичні дослідження.

У той же час, системи підтримки прийняття рішення відрізняються від традиційних систем тим, що включають в себе емпіричну роботу, засновану на досвіді експертів.

У СППР немає конкретного визначення, так як різні дослідники запропонували різні визначення. Одним з них може бути таке. Системи підтримки прийняття рішення неявно спираються на наступні концепції [7]:

– СППР визначається в рамках структури завдання, яке розглядає;

– СППР вимагають особливої стратегії проектування, заснованої на методах еволюції і «середньої ланки»;

– СППР надає когнітивну підтримку процесу для осіб, що приймають рішення. Процес дослідження рішень забезпечує описове розуміння управлінських рішень і нормативну базу, за допомогою яких визначаються способи підвищення ефективності;

– СППР дозволяють менеджерам виконувати управлінські рішення за допомогою комп'ютерів. Дана стратегія досягається при використанні

кваліфікованих експертів, побудованих сервісів і зручного для користувача інтерфейсу.

Іншим визначенням для СППР може служити [8] наступне. Мета системи підтримки прийняття рішення, полягає в підтримці особи, яка приймає рішення, в процесі прийняття рішень.

Також СППР визначається як комп'ютерна програма, яка покращує можливості експерту або групі експертів в процесі прийняття рішень [9].

Іноді СППР визначають [10] як єдину потужну систему, яка підтримує три типи процесів прийняття рішень структурованих, слабо структурованих і неструктурованих проблем. А сама система являє собою комбінацію даних, аналітичних інструментів і призначеного для користувача інтерфейсу.

СППР також визначається як система, яка підвищує якість і ефективність процесу і результатів прийняття рішень [11].

## 1.2 Методи прийняття рішень та їх класифікація

За такими ознаками, як наявність експертної інформації та тип отриманої інформації методи прийняття рішень розділяють на такі групи [12]:

а) коли не потрібна інформація від експерта:

1) метод домінування;

2) метод на основі глобальних критеріїв;

б) Коли потрібна інформація щодо переваг на множині критеріїв:

1) якісна інформація:

– лексикографічне впорядкування;

– порівняння різниць оцінок за критеріями.

2) кількісна оцінка переваг:

– методи «ефективність-вартість»;

– методи згортки на ієрархії критеріїв;

– методи порогів;

– методи ідеальної точки.

3) кількісна інформація про заміщення:

– метод кривих байдужості;

– методи теорії цінності;

в) коли потрібна інформація щодо переваг альтернатив:

1) оцінка переваг парних порівнянь:

– методи математичного програмування;

– лінійна і нелінійна згортка при інтерактивному способі визначення її параметрів.

г) коли потрібна інформація щодо переваг на множині критеріїв та наслідках альтернатив:

1) відсутня інформація щодо переваг (методи з дискретизацією невизначеності);

2) кількісна та/або інтервальна інформація:

– стохастичне домінування;

– методи прийняття рішень в умовах ризику і невизначеності на основі глобальних критеріїв;

3) кількісна інформація о перевагах та наслідках:

– метод аналізу ієрархій;

– методи теорії нечітких множин;

– метод практичного прийняття;

– методи вибору статистично ненадійних рішень.

4) кількісна інформація про переваги та наслідки:

– методи кривих байдужості для прийняття рішень в умовах ризику і невизначеності.

– методи дерев рішень.

Метод аналізу ієрархій (МАІ) є одним зі структурованих прийомів організації та аналізу складних рішень, заснований на математиці та психології.

Для прийняття рішення за допомогою МАІ потрібно розкласти процес рішення на такі етапи [18].

Етап 1. Визначення проблеми та визначення типу необхідних знань.

Етап 2. Структурування ієрархії рішення зверху, починаючи з мети процесу прийняття рішення, потім цілі з більш широкої точки зору, через проміжні рівні (критерії, від яких залежать наступні елементи) до найнижчого рівня (що зазвичай є набором альтернатив).

Етап 3. Побудова набору матриць парних порівняння. Кожен елемент верхнього рівня використовується для порівняння елементів на рівні безпосередньо нижче щодо нього.

Етап 4. Використання пріоритетів, отриманих при порівнянні, для зважування пріоритетів на рівні, що знаходиться нижче. Це робиться для кожного елемента. Потім для кожного елемента в нижньому рівні додаються його зважені значення та отримуються його загальний або глобальний пріоритет. Цей процес зважування та додавання продовжується до тих пір, поки не будуть отримані остаточні пріоритети альтернатив на самому нижньому рівні.

Для порівняння використовується шкала чисел, яка вказує, у скільки разів важливіший або домінуючий один критерій щодо альтернативи, щодо якої проводиться процес прийняття рішення. Приклад такої шкали наведений у табл. 1.

Використовуючи значення з таблиці 1 можливо створити матриці парних порівнянь. Після побудови цих матриць потрібно нормалізувати значення у матрицях [19]. Потім обчислюється внесок кожного критерію відносно мети проведення аналізу, який визначається розрахунками, зробленими за допомогою вектору пріоритету (або власного вектору). Власні вектори показують відносні ваги між кожним критерієм, який він отримує приблизним чином, обчислюючи середнє арифметичне всіх критеріїв.

Наступним кроком є пошук будь-яких неузгоджених даних. Мета полягає в тому, щоб зібрати достатньо інформації, щоб визначити, чи були послідовні особи, які приймають рішення, у своєму виборі. Наприклад, якщо ті, хто приймає рішення, стверджують, що стратегічні критерії важливіші за фінансові критерії та що фінансові критерії важливіші за критерії зобов'язань зацікавлених сторін,

було б непослідовно підтверджувати, що критерії зобов'язань зацікавлених сторін важливіші за стратегічні.

Таблиця 1.1 – Шкала порівняльних оцінок критеріїв

| Ступінь важливості | Протилежне значення | Визначення | Пояснення |
|---|---|---|---|
| 1 | 1 | Однакова важливість | Дві дії однаково сприяють досягненню мети |
| 2 | 1/2 | Не важливий | |
| 3 | 1/3 | Помірне значення | Досвід та судження трохи корисні |
| 4 | 1/4 | Дещо важливий | |
| 5 | 1/5 | Важливий | Досвід та судження дуже сприяють |
| 6 | 1/6 | Більш важливий | |
| 7 | 1/7 | Дуже важливий | Дуже сильно надаються переваги; домінування демонструється на практиці |
| 8 | 1/8 | Вкрай важливий | |
| 9 | 1/9 | Самий важливий | Максимально високий порядок важливості |

Індекс узгодженості (ІУ) базується на максимальній власній величині $\lambda_{max}$, яка обчислюється шляхом підсумовування добутку кожного елемента власного вектора з відповідною сумарним значенням колонки з ненормалізованої матриці порівняння

$$\text{ІУ} = \frac{\lambda_{max} - n}{n - 1}, \tag{1.4}$$

де $n$ – кількість критеріїв;

$\lambda_{max}$ – максимальна власне значення матриці порівнянь.

Для того, щоб перевірити, чи адекватний індекс узгодженості, обчислюється коефіцієнт узгодженості (КУ), який визначається співвідношенням між індексом узгодженості та індексом випадкової консистенції (ВК)

$$КУ = \frac{ІУ}{ВК} \quad (1.5)$$

де ІУ – індекс узгодженості;

ВК – постійна величина, обирається відповідно до кількості критеріїв.

Матриця буде вважатися послідовною, якщо отримане співвідношення менше 10%.

Основна концепція TOPSIS полягає в тому, що обрана альтернатива повинна мати найкоротшу відстань від ідеального рішення і найдальшу від негативно-ідеального рішення (рис. 1.1) [20].

Процес алгоритму TOPSIS [21] починається з формування матриці рішення, що представляє значення задоволеності кожного критерію кожною альтернативою. Далі матриця нормалізується за потрібною схемою нормалізації, а значення множать на критерії ваги. Згодом обчислюються позитивно-ідеальні та негативно-ідеальні рішення, а відстань кожного альтернативи до цих рішень обчислюється за допомогою вимірювання відстані. Потім альтернативи класифікуються на основі їх відносної близькості до ідеального рішення.

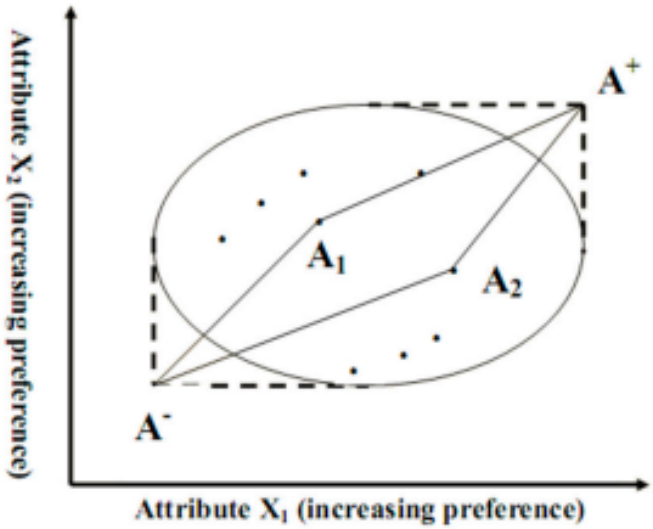


Рисунок 1.1 – Схема базової концепції TOPSIS

Класичний алгоритм TOPSIS передбачає реалізацію семи кроків [20, 21].

Крок 1. Будується матриця рішень та визначаються ваги критеріїв.

Крок 2. Нормалізується матриця рішень.

Крок 3. Обчислюється зважена нормована матриця рішень.

Крок 4. Визнається ідеальні позитивне та негативне рішення. Ідеальне позитивне рішення – це рішення, яке максимізує критерії вигоди та мінімізує критерії витрат, тоді як негативне ідеальне рішення максимізує критерії витрат та мінімізує критерії вигоди. Вони обчислюються за формулами (1.6) та (1.7):

$$A^+ = (v_1^+, v_2^+, \dots, v_n^+) = \left( \left( \max_i v_{ij} \middle| j \in I \right), \left( \min_i v_{ij} \middle| i \in J \right) \right), \quad (1.6)$$

$$A^- = (v_1^-, v_2^-, \dots, v_n^-) = \left( \left( \min_i v_{ij} \middle| j \in I \right), \left( \max_i v_{ij} \middle| i \in J \right) \right). \quad (1.7) \quad (7)$$

де $n$ – кількість критеріїв;

$v_{ij}$ – нормалізоване значення з матриці рішень;

$I$ – множина критеріїв вигоди;

$J$ – множина критеріїв витрат;

Крок 5. Обчислюються міри відокремлення від позитивного ідеального рішення та негативного ідеального рішення.

Крок 6. Обчисліть відносну близькість до позитивного ідеального рішення. Відносна близькість $i$-ої альтернативи $A_j$ щодо $A^+$ обчислюється за формулою (1.8):

$$\boldsymbol{R_i = \frac{d_i^-}{d_i^- + d_i^+}}, \quad (1.8)$$

де $d_i^-$ – міри відокремлення від негативного ідеального рішення;

$d_i^+$ – міри відокремлення від позитивного ідеального рішення.

Крок 7. Оцінюється ранжування уподобання або обирається альтернатива, найближча до 1. Набори альтернативних варіантів можуть бути проранжовані за порядком зменшення значення $R_i$.

TOPSIS є корисним методом для ОПР, коли розглядаються структуруванні проблеми, що вирішуються, проведенням аналізу, порівняннями та ранжируванням альтернатив. Недоліком класичного методу TOPSIS є те, що він вирішує проблеми, в яких всі дані рішення відомі і представлені чіткими числами, однак більшість сучасних проблем мають більш складнішу структуру.

Метод SAW вимагає від особи, яка приймає рішення, визначати вагу кожного атрибута. Загальний бал за атрибутом отримується шляхом підсумовування всіх результатів номінального множення та ваги кожного атрибута. Рейтинг кожного атрибута повинен бути безрозмірним у тому сенсі, що він пройшов процес нормалізації попередньої матриці [22].

Метод SAW часто також відомий як метод зваженого підсумовування. Основна концепція методу SAW полягає у пошуку зваженої суми оцінок ефективності для кожної альтернативи за всіма ознаками. Необхідно нормалізувати матрицю рішень до шкали, порівнянної з усіма існуючими альтернативними оцінюваннями:

$$\boldsymbol{R_{ij}} = \begin{cases} \dfrac{\boldsymbol{x_{ij}}}{\boldsymbol{max\, x_{ij}}}, \\ \dfrac{\boldsymbol{min\, x_{ij}}}{\boldsymbol{x_{ij}}}, \end{cases} \tag{1.9}$$

де $x_{ij}$ – рядки та стовпці матриці;

$max\, x_{ij}$ – максимальне значення кожного рядка та стовпця;

$min\, x_{ij}$ – мінімальне значення кожного рядка та стовпця.

Значення переваг для кожної альтернативи обчислюється за формулою (1.10):

$$\boldsymbol{V_i} = \sum_{\boldsymbol{j=1}}^{\boldsymbol{n}} \boldsymbol{w_j r_{ij}}, \tag{1.10}$$

де $V_i$ – кінцеве значення альтернативи;

$w_j$ – ваговий коефіцієнт;

$r_{ij}$ – нормалізоване значення.

Переваги методу SAW в його здатності робити оцінку точніше, оскільки він заснований на заздалегідь визначених критеріях та перевагах.

Однак SAW ряд має недоліків [23]:

– усі значення критеріїв $R_{ij}$ повинні бути визначені для умов максимізації. Критерії мінімізації, перш ніж використовуватись в аналізі, повинні бути перетворені на критерії максимізації;

– усі значення критеріїв $R_{ij}$ повинні бути додатними;

– оцінки, отримані SAW, не завжди відображають реальну ситуацію. Отриманий результат може бути нелогічним, оскільки значення одного конкретного критерію значною мірою відрізняються від значень інших критеріїв.

## 1.3 Методи формування підмножини ефективних рішень

Множина альтернативних рішень звужується до множини допустимих рішень на основі врахування обмежень. Допустимими, або прийнятними, називаються рішення, що задовольняють множині встановлених обмежень. Процедура отримання множини прийнятних рішень з початкової множини може виконуватися шляхом логічного мислення або формально, в залежності від ступеня формалізації інформації.

Звуження множини допустимих рішень до множини ефективних рішень здійснюється на основі аналізу переваг. Рішення називається ефективним, якщо не існує більш пріоритетного. Множину ефективних рішень в літературі називають також множиною Парето, множиною недомінованих рішень. Усі ефективні рішення між собою непорівнянні, тобто не можна сказати, яке з них краще. В окремих випадках множина ефективних рішень може містити тільки одне рішення або збігатися з множиною допустимих рішень.

У загальному випадку в процесі проектування складних об’єктів здійснюється розв’язання взаємопов’язаних задач структурної, топологічної,

параметричної оптимізації тощо. внаслідок NP-складності таких задач здійснюються генерація й аналіз великої кількості варіантів проектних рішень $X$. При цьому лише невелика кількість серед них є ефективними (оптимальними за Парето) за множиною показників якості:

$$X^S \cup X^K, Card(X) >> Card(X^K), \tag{1.11}$$

де $X^S$– множина неефективних рішень, будь-яке з яких може бути поліпшене хоча б за одним показником якості без погіршення якості за іншими показниками;

$X^K$– множина компромісів (ефективних рішень), ні одне з яких не може бути покращене одночасно за всіма показниками якості.

Відомі точні методи розв'язання задач виділення підмножин ефективних рішень $X^K$ (1.11) мають високу часову складність. Це обумовлює актуальність задач визначення підмножин ефективних варіантів, серед яких проектувальники здійснюють остаточний вибір [24, 25].

Для розв'язання задачі необхідно удосконалити метод та розробити засіб, у якому була б передбачена можливість вибору найкращого методу, виходячи з особливостей множин допустимих проектних рішень.

При цьому для множин допустимих рішень $X$ відносно невеликих потужностей доцільно використовувати метод парних порівнянь усіх можливих пар розв'язань $x_i, x_j \in X$, тобто $x_1$ і $x_2$, $x_1$ і $x_3$,…, $x_2$ і $x_3$, $x_2$ і $x_4$,… та видаленні з подальшого розгляду рішень, які за всіма частковими критеріями гірші за інші.

Для опуклих множин проектних варіантів великої потужності доцільно використовувати метод на основі теореми Карліна [26].

З його використанням підмножина $X^K$ знаходиться об'єднанням варіантів рішень $x_i^0, i = \overline{1,m}$, що оптимізують кожен з локальних показників якості $k_i(x), i = \overline{1,m}$, з розв'язання відносно параметрів задачі:

$$\boldsymbol{\lambda_i \in \Lambda = \left\{ \lambda_i : \lambda_i > 0 \; \forall i = \overline{1,m} \; \sum_{i=1}^{m} \lambda_i = 1 \right\},}$$
$$\boldsymbol{x_i^o = \arg\max_{x \epsilon X} \{ P(x) = \sum_{i=1}^{m} \lambda_i \cdot \bar{k}_\iota(x) \}} \quad (1.12)$$

де $\bar{k}_\iota(s)$ – значення функції корисності часткового показника якості (критерію) $k_i$, $i = \overline{1,m}$ або його нормоване значення.

Для великої кількості варіантів, що утворюють неопуклу множину, можна використати метод, що побудований на основі теореми Гермейєра [26]. У цьому випадку підмножина $X^K$ може бути знайдена об'єднанням варіантів $x_i^0, i = \overline{1,m}$, що оптимізують кожен з показників якості, $k_i(x), i = \overline{1,m}$, з розв'язання задачі відносно параметрів:

$$\boldsymbol{\lambda_i \in \Lambda = \left\{ \lambda_i : \lambda_i > 0 \; \forall i = \overline{1,m} \; \sum_{i=1}^{m} \lambda_i = 1 \right\},}$$
$$\boldsymbol{x_i^o = \arg\max_{x \epsilon X} \{ P(x) = \min \lambda_i \cdot \bar{k}_\iota(x) \}} \quad (1.13)$$

Метод парних порівнянь є універсальним, проте має відносно високу часову складність. Часова складність і точність методів Карліна та Гермейєра визначаються кроком розв'язання задач (1.12), (1.13). Вибір методу та кроку розв'язання задач (1.12) або (1.13) на практиці буде визначатися обчислювальними ресурсами системи автоматизації проектування.

## 1.4 Постановка мети та завдань дослідження

Ефективність об'єктів, що проектуються для використання в різних сферах діяльності, суттєво визначається рішеннями, які приймаються уже в процесі їх проектування. У процесі проектування передбачається ітераційне розв'язання множини завдань технологічної, структурної, параметричної, топологічної

оптимізації за функціональними та вартісними показниками (критеріями ефективності). Вибір найкращого варіанту з множини допустимих тільки лише в найпростіших ситуаціях здійснюється ОПР. В наслідок комбінаторного характеру завдань синтезу кількість варіантів проектних рішень різко збільшується з ростом розмірності задач проектування.

Зазвичай переважна більшість варіантів проектів є неефективними (домінованими). Кожен з таких проектів може бути поліпшений одночасно за всіма показниками. Виникає проблема виділення чи формування підмножини тільки Парето-оптимальних (ефективних) проектних рішень.

Метою даної роботи є підвищення ефективності систем підтримки прийняття багатокритеріальних проектних рішень за рахунок удосконалення процедур виділення підмножин ефективних варіантів.

Для досягнення поставленої мети необхідно вирішити такі завдання:

– провести аналіз сучасного стану проблеми підтримки прийняття проектних рішень;

– обрати базові методи виділення підмножин ефективних проектних рішень;

– удосконалити або модифікувати базовий метод виділення підмножин ефективних проектних рішень;

– програмно реалізувати удосконалений (модифікований) метод виділення підмножин ефективних проектних рішень;

– провести експериментальне порівняльне дослідження ефективності удосконаленого (модифікованого) методу виділення підмножин ефективних проектних рішень;

– надати рекомендації щодо практичного використання удосконалених (модифікованих) методів виділення підмножин ефективних проектних рішень у системах підтримки прийняття проектних рішень.

Об’єктом досліджень є процес підтримки прийняття проектних рішень.

Предметом досліджень є процедури виділення підмножин ефективних варіантів у системах підтримки прийняття проектних рішень.

# 2 МЕТОДИ ВИЗНАЧЕННЯ ПІДМНОЖИН ЕФЕКТИВНИХ ВАРІАНТІВ

## 2.1 Метод попарних порівнянь

Цей підхід стосується узагальнення суджень попарних порівнянь від групи осіб, що приймають рішення (ОПР). Попарні порівняння дозволяють розглядати лише параметри елементів рішення та визначати їх перевагу та ступінь переваги між парою нематеріальних коефіцієнтів.

Подібна сегментація проблеми рішення досягається за допомогою Закону порівняльного судження. Наприклад, є два елементи $x$ та $y$ , визначається, що надається перевага елементу $x$ над елементом $y$ з позначенням $x \succ y$. Різні числові шкали можуть бути використані для відображення сили переваги; найбільш широко використовуваною є шкала Сааті [27].

Загалом, проблема прийняття рішення [28] може бути сформульована наступним чином: є множина альтернатив розмірністю, $X = \{x_1, x_2, \ldots, x_n\}$ (об'єкти, особи, критерії прийняття рішень), які слід класифікувати від кращих до найгірше, або навпаки, спираючись на інформацію, подану в матриці парних порівнянь, $A = \{a_{ij}\}$ .

Найважливішим кроком у процесі прийняття рішення є визначення зважених ваг, тобто ваги для множини $X$ альтернатив відносно критеріїв або експертних оцінок. Спосіб визначення ваг – починається з відношення, представленого матрицею парних порівнянь $A = \{a_{ij}\}$; кожен елемент цієї матриці $a_{ij}$ – це додатне дійсне число, яке ступінь переваги $x_i$ над $x_j$..

Елементи $a_{ij}$ з матриці парних порівнянь $A = \{a_{ij}\}$ можуть бути взяті зі шкали $S$ у залежності від проблеми прийняття рішення, наприклад:

– $S = \{0,1\}$ – бінарна шкала;

– $S = \{{}^1/_9, {}^1/_8, \ldots, {}^1/_2, 1,2, \ldots, 9\}$ – шкала MAI;

– $S = ]0; +\infty[$, або $S = ]-\infty; +\infty[$ – інтервальна шкала;

– $S = [0;\ 1]$ – одинична інтервальна шкала;

– $S =$ масштаб відкритого інтервалу, при використанні груповою операцією.

Матриця парних порівнянь, як розглядається у цій роботі має наступний вигляд:

$$\begin{array}{cccc} & X_1 & \cdots & X_n \\ A_1 & x_{11} & \cdots & x_{1n} \\ \vdots & \vdots & \ddots & \vdots \\ A_m & x_{m1} & \cdots & x_{mn} \end{array}$$

де $A_m$ – альтернатива $n$,

$m$ – кількість альтернатив,

$X_n$ - критерій $n$,

$n$ – кількість критеріїв,

$x_{mn}$ – значення критерію $n$ для альтернативі $m$.

У загальному випадку питання побудови множини недомінуючих рішень або векторів представляється надзвичайно складним, однак для кінцевої множини можливих рішень X (множини можливих векторів Y) він вирішується досить просто.

Перед тим, як запровадити метод, спочатку, потрібно пронумерувати можливі рішення, з яких складається первинна множина, наприклад $X = X_1 = \{x_1, x_2, \dots, x_n\}$. Ще однією умовою використання методу є те, що множина має бути кінцевим та відносини переваги мають бути транзитивними та ірреflексивними [29].

Загальний перелік кроків алгоритму має наступний вигляд [29] (рис. 2.1):

а) проводиться послідовне порівнянні рішення $x_1$ з усіма, що залишились $x_2, \dots, x_n$. У процесі порівнянь перевіряється справедливість співвідношень $x_i \succ x_1$ та $x_1 \succ x_i$ , де $i = 2, \dots, n$.

Якщо для деякого $i$ з $x_1 \succ x_i$ співвідношення є істинним, то домінуюче рішення $x_i$ виключається з множини $X$ та продовжити виконувати зазначену операцію для наступного рішення $x_i$.

Якщо виконується співвідношення $x_i \succ x_1$, тоді потрібно виключити з множини рішення $x_1$, а також одразу перейти до наступного кроку методу.

Якщо жодне з співвідношень $x_i \succ x_1$ та $x_1 \succ x_i$ не виконується, то ніяке рішення не буде вилучене з множини $X$.

Якщо при порівнянні рішення $x_1$ з усіма іншими рішеннями $x_2, \dots, x_n$, та для ніякого $i = 2, \dots, n$ не виконується $x_i \succ x_1$, рішення можна вважати як недомінуюче та виключити його з множини можливих рішень.

Важливо, відмітити, що якщо після першого кроку виконання методу в множині можливих рішень усі рішення є видаленими, то алгоритм закінчує роботу. Тоді множина недомінуючих рішень складається лише з рішення $x_1$.

б) другий крок методу є аналогічним першому. Спочатку нумеруються рішення у множині $X_2$, яка відповідає множини, яка залишилась після виконання першого кроку. Після проводиться послідовне порівняння $x_1$ з $X_2$ з іншими рішеннями. Результатом виконання операцій порівнянь на другому кроці може бути або видалення домінуючого першого рішення з множини, або видалення не відбудеться. У такому разі це рішення можна виділити як не домінуюче, що призводить до його видалення з множини $X_2$. Якщо після жодного рішення не залишиться у множині, тоді можна припинити роботу алгоритму, тому, що множина не домінуючих рішень знайдена. У противному випадку алгоритм продовжує свою роботу аналогічно першому та другому кроку.

В результаті роботи алгоритм буде знайдена множина всіх не домінуючих рішень. Схематичне представлення алгоритму наведено на рис 2.1.

Одним з недоліків цього методу можна вважати велику часову складність. Так як треба провести порівняння усіх альтернатив між собою при збільшенні кількості їх кількості час буде рости за експонентною.

До переваг даного методу можливо віднести те, що усі можливі ефективні рішення будуть знайдені у процесі роботи алгоритму. Однак, якщо первинна множина складається з великої кількості (більш 1000) у множині не домінуючих може оказатися доволі багато рішень, що ускладнює для ОПР або групи ОПР процес прийняття рішення.

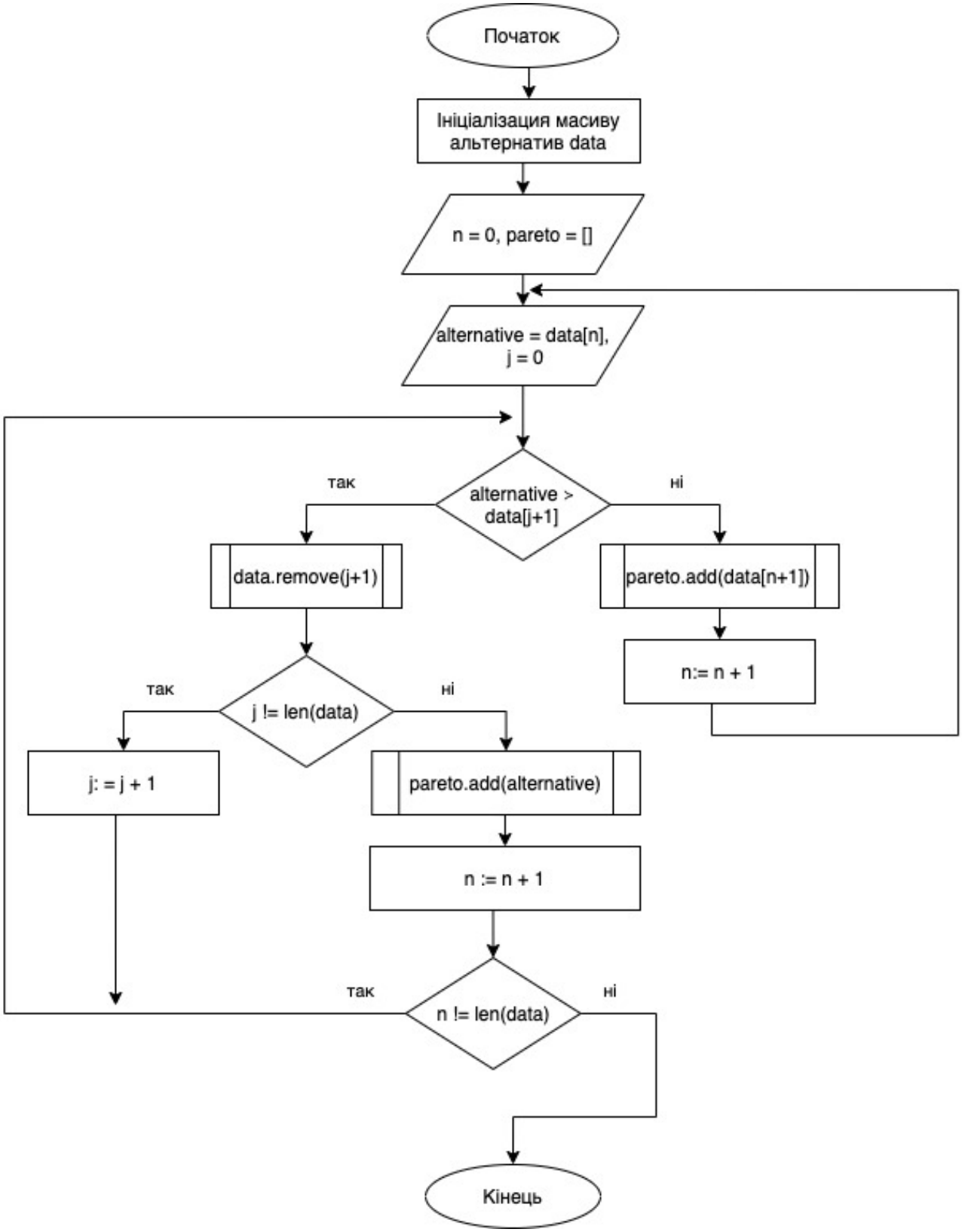


Рисунок 2.1 – Схема алгоритму попарних порівнянь

## 2.2 Визначення наближеної множини компромісів

Як було позначено раніше деякі методи відзначає велика часова складність, однією з можливостей подолання цієї проблеми є побудова наближеної множини компромісів $S^P$, умовою для цього є те, що $S^K \subseteq S^P$, де $S^K$ – підмножина ефективних варіантів [30].

Існує декілька методів  побудова наближеної множини компромісів $S^P$, наприклад, можна використовувати методи виділення сектора і сегмента $S^K \subseteq S^P \subseteq S^*$ [30] та метод роя частинок.

Загалом, процес побудови наближеної множин компромісів (НМК) (рис. 2.2) [31] можливо сформулювати наступним чином, послідовно розв'язуються $n$ оптимізаційних задач за одним критерієм на множині допустимих рішень по кожному частинному критерію $k_i(x), i = \overline{1,n}$, тобто:

$$x_i^0 = arg \max_{x \in X} k_i(x)$$

Для визначення НМК потрібно сформувати таблицю [31], кожен рядок якої містить значення частинного критерію $i$ в точках екстремум за всім критеріями. Голова діагональ містить екстремуми критеріїв.

Для цього потребує прийняти, що $\alpha$ дорівнює одиниці, тоді відповідно до моделі всі інші коефіцієнти дорівнюють нулю:

$$X^C = \bigcup_{a \in A} arg \max_{x \in X} a_i k_i(x),$$

де $A = \begin{cases} 0 \le a_i \le 1 \\ \sum_{i=1}^{n} a_i = 1, \forall i = \overline{1,n} \end{cases}$,

$a = \langle a_1, a_2, \dots, a_n \rangle$ – вагові коефіцієнти критеріїв.

За цих умов може буде отримана таблиця (табл. 2.1), яка буде зберігати результати для наближеної множини (області) компромісів. Розрахунок значення кожної клітинки зберігає у себе значення частинних критеріїв $\mathrm{k_j}(\mathrm{x_i^0}) = \mathrm{k_{ij}}, \mathrm{j} = \overline{1, \mathrm{n}}$ для кожного рішення $x_i^0$. Це означає, що у кожному рядку значення критерію змінюється від найкращого до найгіршого, множина цих значень будуть межами наближеної області компромісів на множині критеріїв $K = \{k_i(x)\}$ за виконанням обмежень $x \in X, k^- \leq k_i(x) \leq k^+, i = \overline{1, n}$ .

Таблиця 2.1 – Дані для визначення границь наближеної множини компромісів

| $k_i(x), i = \overline{1, n}$ | $k_1(x)$ | $k_2(x)$ | ⋯ | $k_n(x)$ |
|---|---|---|---|---|
| $k_1(x)$ | $k_{11}$ | $k_{12}$ | ⋯ | $k_{1n}$ |
| $k_2(x)$ | $k_{21}$ | $k_{22}$ | ⋯ | $k_{21}$ |
| ⋮ | ⋮ | ⋮ | ⋯ | ⋮ |
| $k_n(x)$ | $k_{n1}$ | $k_{n2}$ | ⋯ | $k_{nn}$ |

Недоліком цього методу є те, що необхідно перевіряти компромісні рішення, вибрані з наближеною області на приналежність до області Парето. Він не є визначальним. Однак, перевагою цього методу є можливість значно зменшити область аналізу при пошуку компромісного рішення та перевірки приналежності рішення до області Парето. Ще однією перевагою є спрощення нормування часткових критеріїв [32].

Ще одним алгоритм визначення наближеної множини компромісі в є метод рою частинок. Оптимізація багатокритеріальних задач цим методом є популярним алгоритмом, який широко застосовується для вирішення одно- та багатокритеріальних задач протягом останніх двох десятиліть. Оптимізація рою частинок включає кілька локальних та глобальних стратегій пошуку, а також стратегії навчання та адаптації параметрів для того, щоб покращити її ефективність протягом багатьох років [33].

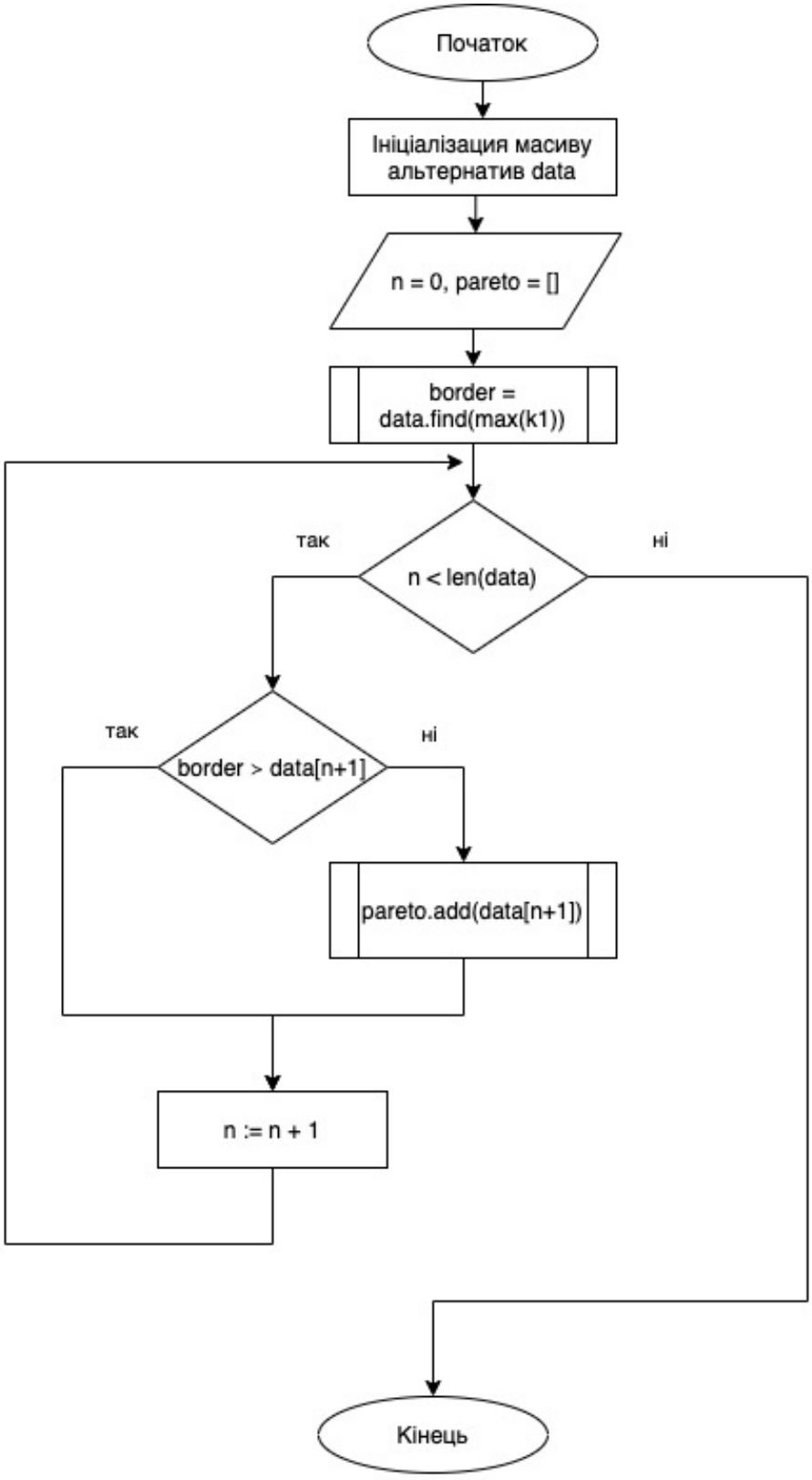


Рисунок 2.2 – Схема алгоритму виділення наближеної множини компромісів

Цей алгоритм не лише використовується як самостійний алгоритм, а також використовується у ланці еволюційних алгоритмів.

Алгоритми рою частинок оновлює швидкість і розташування частинок за допомогою двох «напрямних»: оптимальне рішення, яке знайшла частинка (тобто направляюча індивідуума $\overrightarrow{p_{best}}$) і оптимальне рішення, яке до цього часу знаходило уся популяція (тобто глобальна направляюча $\overrightarrow{g_{best}}$). Оновлення швидкості та розташування задовольняє наступним умовам [34]:

$$v_{id}(t+1) = \omega v_{id}(t) + r_1 c_1\big(p_{id} - x_{id}(t)\big) + r_2 c_2\big(g_d - x_{id}(t)\big)$$
$$x_{id}(t+1) = x_{id}(t) + v_{id}(t+1)$$

де у d-мірному напрямку пошуку:

$v_{id}(t)$ – швидкість частинки,

$x_{id}(t)$ – розташування частинки,

$v_{id}(t+1)$ – оновлена швидкість частинки,

$x_{id}(t+1)$ – оновлене місце розташування частинки,

$p_{id}$ – оптимальне рішення, яке знайшла частинка,

$g_{id}$ – оптимальним рішенням, яке знайшла до цього часу вся популяція,

$\omega$ – вага інерції,

$c_1$ і $c_2$ – коефіцієнти прискорення,

$r_1$ і $r$ – випадкові числа в інтервалі.

Недоліком цього методу є те [33], що більшість із цих підходів призводить до збільшення кількості параметрів, які треба визначити користувачу або ОПР, та алгоритмічних кроків, що призводить до збільшення складності алгоритму.

У даній роботі розглядається перший метод визначення наближеної множини компромісів, так як цей метод у порівнянні з методом рою частинок має меншу алгоритмічну складність та є більш незалежним від суб’єктивних факторів. Схема алгоритму представлена на рис. 2.2.

## 2.3 Лінійна згортка та лема Карліна

Метод лінійної згортки критеріїв – є одним з найвідомішим і найпоширенішим при вирішенні оптимізаційних багатокритеріальних задач. Він полягає в призначенні коефіцієнтів в лінійній згортці вихідних критеріїв та подальшого зведення її до екстремуму на множині допустимих варіантів. Знайдене таким чином рішення у рамках цього методу може вважатися «найкращим» [35].

Для застосування адитивної або лінійної згортки критеріїв необхідною умовою є, засноване на теоремі Карліна (лема Карліна), а також достатні умови Парето-оптимальності [36].

В якості альтернативи побудови кривих нелінійних функцій корисності може використовуватися [37] лінійна згортка критеріїв $U(y) = \sum_i c_i y_i$ коефіцієнти $c_i$ якої (ваги критеріїв) повинен вказати ЛПР. Подібні функції ефективно застосовуються для опуклих задач.

У рамках багатокритеріальних задач використання лінійної згортки критеріїв є свідомим, якщо виконується аксіома виключення домінуючих векторів, а також якщо існують коефіцієнти $c_1, c_2, \dots, c_m$ при яких відношення переваги $\succ$ є лінійної функцією, тобто:

$$y \succ y` \Leftrightarrow \sum_{i=1}^{m} \mu_i y_i > \sum_{i=1}^{m} \mu_i y_i`,$$

для всіх $y, y` \in R^m$.

Це означає, що будь-який вектор, для якого знайдеться інший вектор з більшою лінійною комбінацією буде виключений з множини. Тобто у множині залишаться тільки ті, які доставляють максимум лінійній згортці [35].

Лема Карліна може бути сформульована, так. Нехай множина варіантів $X \subset R^n$ опукла, а функції $f_1, .., f_n$ увігнуті на ньому. Якщо $x^* \in P_j(X)$, то існує набір чисел $\lambda_i \geq 0, \sum_{i=1}^{m} \lambda_i = 1$, такий, що лінійна згортка $\lambda_i \geq$

$0, \sum_{i=1}^{m} \lambda_i = 1$ з зазначеними коефіцієнтами досягає максимуму в точці $x^*$ на множині $X$ .

У роботі [38] розглядається, що істотним для даної леми є умова не заперечності коефіцієнтів, опуклості множини й угнутості критеріїв. Доведено, що деякі оптимальні по Парето варіанти не будуть обрані (навіть при умові завдання експертом негативних коефіцієнтів), тоді як згідно з принципом Еджворта-Парето обраним може виявитися будь-який елемент з $P_j(X)$. Подібна ситуація виникає при використанні МАІ для неопуклої множини $X$. З цього випливає, що метод лінійної згортки можуть призводити до виборів неоптимальних рішень. Подібна ситуація виникає при використанні МАІ для неопуклої множини $X$. З цього випливає, що метод лінійної згортки можуть призводити до виборів не оптимальних рішень. Стверджується, що замість лінійної згортки критеріїв можна використовувати узагальнений критерій $\Phi_\mu\big(f_1(x), \dots, f_m(x)\big)$, де може бути присутнім вектор параметрів $\mu$ певної розмірності, не обов'язково збігається з $m$ . Крапку максимуму узагальненого критерію на множині X стає найкращим варіантом. Кожне необхідне достатнє умови оптимальності по Парето) можна взяти за основу для методу узагальненого критерію в різних варіаціях.

Недоліком лінійної згортки [37] є те, що надмірне значення одного з критеріїв може компенсувати значення іншого.

Однак вважається, що в теорії багатокритеріальної корисності, де ваги критеріїв описуються увігнутими функціями корисності недостатньо велике значення критерію за частіше супроводжується високим ступенем важливості. Подібне може означати низьку збалансованість такого набору критеріїв.

Однак при використанні лінійної функції мінімальне значення одного критерію завжди можна компенсувати надлишковим значенням іншого.

У роботі [39] задля подолання недоліків використання лінійної згортки для звуження множини Парето пропонується використовувати модифікований квадратичний критерій якості.

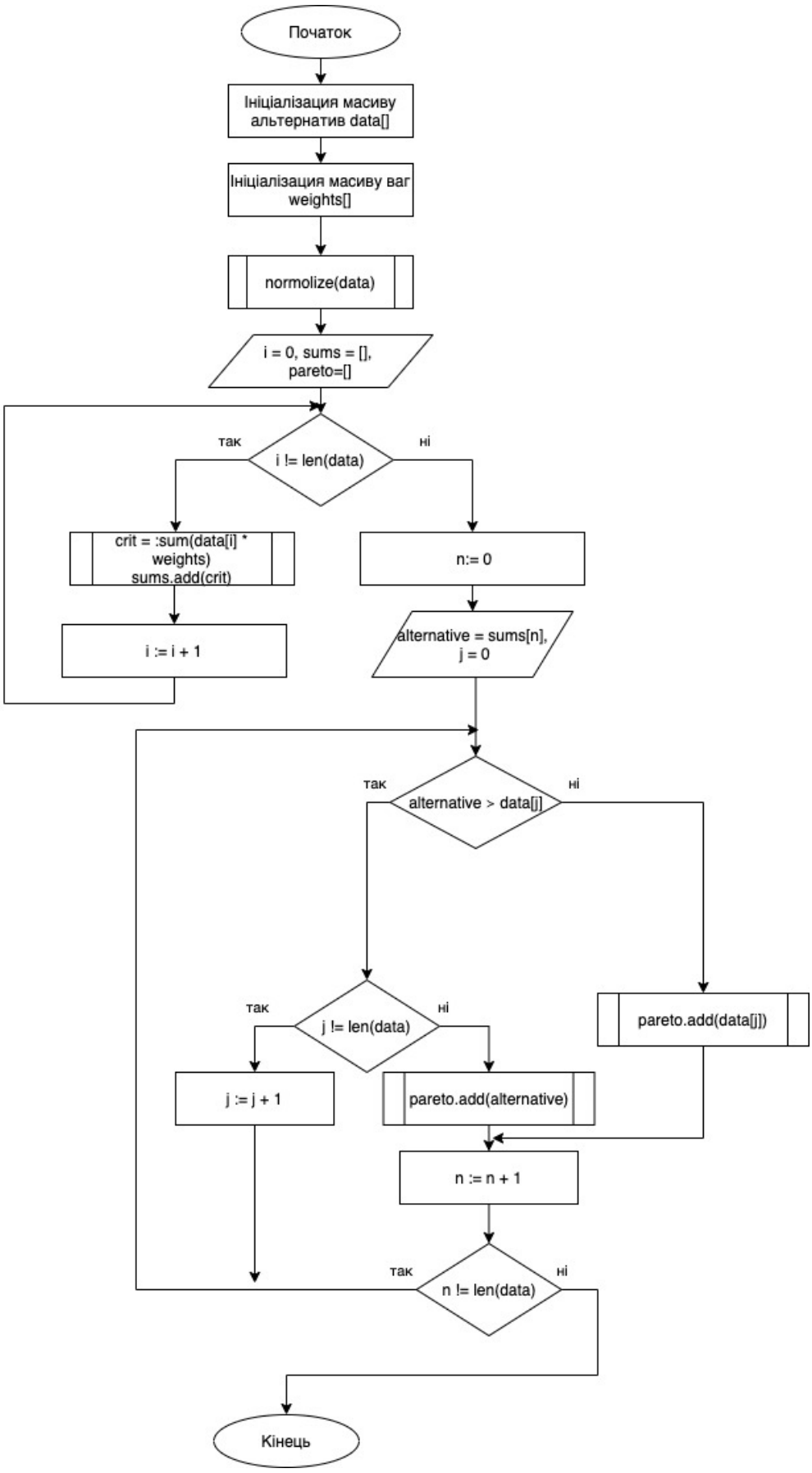


Рисунок 2.3 – Схема алгоритму методу Карліна

У задачі з двома критеріями $J_1$ та $J_2$, у якості вхідних даних вони подаються у вигляді інтеграла від фазових координат:

$$J_1(u) = \frac{1}{2}\int_{t_0}^{t_k} [x^T(t)Qx(t)]dt.$$

Інтеграл від витрат на управління:

$$J_2(u) = \frac{1}{2}\int_{t_0}^{t_k} [u^T(t)Ru(t)]dt.$$

Вагові коефіцієнти подаються у вигляді діагональних матриць $Q = \left(q_{ij}\right)_{n\times n}$ та $R = \left(r_{ij}\right)_{n\times n}$ після обчислення інтегралів були отримані наступні співвідношення:

$$x^T(t)Qx(t) = q_1x_1^2 + q_2x_2^2 + \cdots + q_nx_n^2$$
$$u^T(t)Ru(t) = r_1u_1^2 + r_2u_2^2 + \cdots + r_nu_n^2$$

Авторами пропонується побудова фронтів Парето визначає через вагове співвідношення між витратами на управління та штрафами за відхилення від фазових координат та використання допоміжного масштабування $\sigma$ між групами вагових коефіцієнтів $q_1 = \sigma r_i$.

$$q_{ij} = \left(\frac{x_{1max}}{x_{i\,max}}\right)^2 q_{11},$$
$$r_{ij} = \left(\frac{u_{1max}}{u_{i\,max}}\right)^2 r_{11}.$$

Завдяки цьому були отримані модифіковані інтеграли, залежність яких дозволяє отримати компромісні рішення за допомогою вирішення задачі синтезу

для об’єднаного критерію, де внутрішні коефіцієнти $F_2, \dots, F_n$ та $U_2, \dots, U_n$ масштабують зсередини кожний з критеріїв:

$$J_1'(u) = \frac{1}{2}\int_{t_0}^{t_k} [x_1^2 + F_2 x_2^2 + \cdots + F_n x_n^2]dt,$$

$$J_2(u) = \frac{1}{2}\int_{t_0}^{t_k} [u_1^2 + U_2 u_2^2 + \cdots + U_m u_m^2]dt.$$

Така методика дозволяє отримати при побудові Парето-фронтів оптимальні рішення, які у перехідному процесі не показують майже ніякого погіршення, а також виграють у витратах на управління.

Недоліком подібної методики можливо вважати те, що відхилення тісно пов’язанні з принципом рівноважного вкладу.

Недоліком вищезазначеного методу багатокритеріальної оптимізації є те, що він  вимагають додаткового вкладу від особи, яка приймає рішення, що частіше є суб'єктивним [40].

Перевагами даного методу є те, що як і більшість класичних методів, він має дещо простою структуру і легко розширюється,  іншого боку правильно вибрати параметри уявляється важкою задачею. Хоча метод зваженої суми є найпростішим і найпростішим способом отримання декількох точок на оптимальному для Парето фронті, він не працює в пошуку увігнутих (лінійних) оптимальних витрат. Важко вибрати зважування, щоб забезпечити рівномірний розподіл очок, а також важко знайти всю оптимальну вартість, змінивши ваги [40].

## 2.4 Згортка Гермейєра та зважена метрика Чебишева

Іншою альтернативою лінійної згортки критеріїв є мінімаксна згортка Гермейєра, який дозволяє виявити більш ефективних рішень на множині. Вона визначається наступним співвідношенням [41]:

$$\Psi_\lambda(y) = max\{\lambda_i f_i(y): i = 1, \dots, n\}, \lambda_i \geq 0 \sum_{i=1}^{n} \lambda_i = 1$$

Умовою використання даного методу [41, 42] є додатні значення усіх компонентів векторного критерії у точці $f(y^*) > 0_n$. Ця умова також зумовлює головний недолік цієї згортки – додатність усіх критеріїв у області їх визначення.

Ознакою цієї згортки може вважатися те що за рахунок присвоєння кожному критерію своєї ймовірності досягається певний рівень еластичності. Орієнтований на величини втрат, тобто на негативні значення.

Якщо усі коефіцієнти мають однакове значення, то згортка Гермейєра має точне співпадіння з точність до постійного множника з зваженою метрикою Чебишева.

Визначається [43], що більшість скалярних методів не знаходить усі оптимальні рішення, винятком може бути скалярна функція Чебишева, яка може бути використана для опуклих чи невипуклих множин, гарантуючи при цьому рішення, що є принаймні слабо оптимальними за Парето. Окрім цього, існує теорема нелінійної багатокритеріальної оптимізація, яка при сумісному використанні з функцією Чебишева, може допомогти знайти всі оптимальні рішення Парето для деякого зваженого вектору.

Функція Чебишева має наступний вид:

$$f(x) = \max_i \omega_i |f_i(x) - z_i|,$$

де $z_i < \min_{x \in \Omega} f_i(x), i = 1, \dots, k.$

Недоліком цього методу є те, що сприймається спроба визначити увесь фронт Парето, але також означає, що напрямок пошуку в об'єктивному просторі не є постійним.

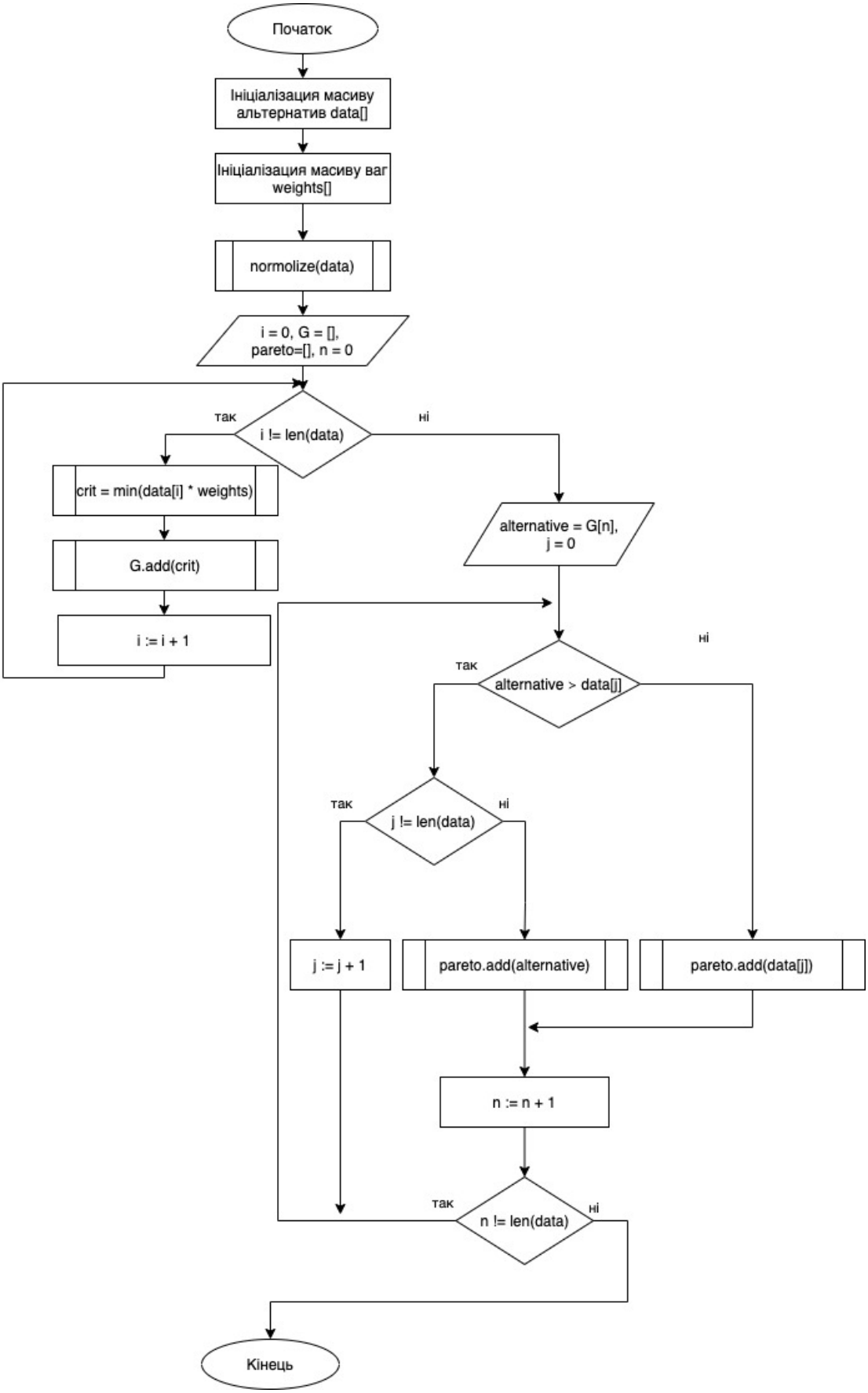


Рисунок 2.4 – Схема алгоритму методу згортки Гермейєра

Також до недоліків додають те, що $f(x)$ не можливо диференціювати, що робить не можливим рішення градієнтним методом однокритеріальної задачі.

Як і лінійна згортка, метод Гермейєра та Чебишева спирається на суб'єктивні оцінки, отримані від ОПР, для визначення ваг. Що затрудняє процес автоматизації пошуку підмножини ефективних варіантів.

Ознакою до використання цього методу є інформація про геометрію фронту Парето, що дозволить знайти оптимальну функцію масштабування. Тоді ймовірність отримання кращого рішення стосовно певної скаляризуючої функції зменшується повільніше порівняно з усіма іншими параметрами масштабу функції, які можуть забезпечити однакову гарантію пошуку всіх оптимальних рішень Парето.

У рамках дослідження було розглянуто реалізацію алгоритму (рис. 2.4) для задачі звуження підмножини ефективних рішень з умовою рівності ваг часткових критеріїв.

## 2.5 Алгоритм NSGA-II

Алгоритм NSGA-II реалізує еволюційний метод виділення підмножини ефективних варіантів. Він використовує елітарний принцип, тобто елітам популяції надається можливість бути перенесеними у наступне покоління; використовує явний механізм збереження різноманітності (відстань переповнення); визначає непереважаючі рішення.

Перевагою [44] цього алгоритму над іншими з сімейству подібних генетичних алгоритмів є те, що була знижена обчислювальна складність алгоритму. Раніше вона складала $O(MN^3)$ (де M – кількість критеріїв и N – розмір популяції).

Загальний алгоритм NSGA-II може бути поданим у наступному виді (рис. 2.1):

– виконується недомінуюче сортування на сукупній популяції (поєднання популяцій батьків та нащадків) та розподілення їх за фронтами;

– заповняється нова популяція відповідно до фронтових рангів;

– якщо один фронт попадає до наступної популяції частково, то виконується сортування, яке використовує відстань, що пов'язана з щільністю розчинів навколо кожного рішення. Кращими індивідуумом вважається той, що має меншу щільність;

– створюється потомство з нової популяції, використовуючи переповнений турнірний вибір (порівнюється за ранговим рейтингом, якщо рівним, то за відстані переповнення), операторами схрещення та мутацій.

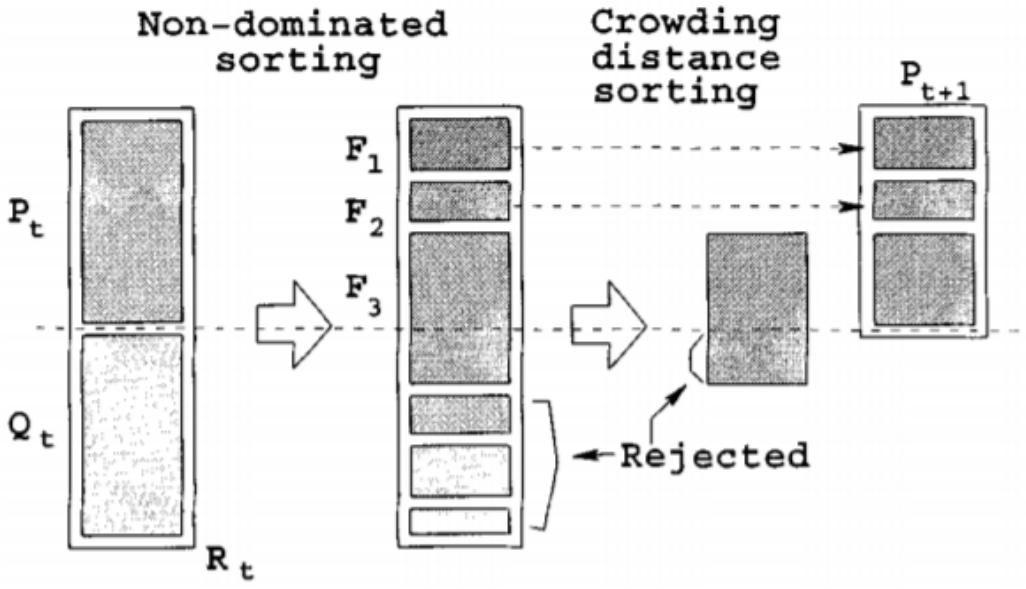


Рисунок 2.1 – Схема роботи алгоритму NSGA-II

Для j-го у $rank_i$ його відстань скупчення (позначена $d^j$), де $m$ – кількість критеріїв, визначається наступним чином: спочатку відстань скупчення $d^j$ дорівнює нулю. Для кожної цільової функції $k$ проводиться сортування індивідуумів в $rank_i$. Якщо для j-го індивідуума цільова функція є граничною (мінімум чи максимум), то $d^j = inf$ . В інших випадках, сортування для для j-го індивідуума порядку $n$ :

$$d^j = d^j + \frac{f_{n+1}^k - f_{n-1}^k}{f_{max}^k - f_{min}^k}.$$

За використання подібного підходу індивідуум, який знаходиться на границі завжди обирається.

NSGA-II використовує сортування без домінування за значенням фітнес функції.

Всім особам, у яких не домінують на іншими особами, присвоюється Парето фронт номер один. Усім особам, які домінують над особами з першого фронту, присвоюється фронт Парето два і так проводиться поки усі особи не будуть співвідноситись з якимось Парето фронтом. Подальший відбір проводиться з використанням турніру між двома особами. Особа з найнижчого фронту обирається, якщо дві особини з різних фронтів. Особа з найбільшою дистанцією скупчення вибирається, якщо вони з одного фронту. Це означає, що особа з більш високою придатністю, розташована на малонаселеній частині фронту. В кожній ітерації є $N$ батьків і народжується $N$ нових особин (потомство). І батьки, і нащадки змагаються між собою за включення до наступної покоління [45].

У NSGA-II використовуються бінарні схрещення (SBX) та поліноміальна мутація.

Оператор SBX працює з двома батьківськими рішеннями та створює два нащадки. Різниця між нащадком та батьком залежить від індексу схрещення $\eta_c$. Індекс схрещення завжди буде невід'ємне дійсне число. Велике значення $\eta_c$ дає більшу ймовірність створення "близько батьківських" рішень, а невелике значення $\eta_c$ дозволяє віддаленим рішенням вибиратись як потомство. Два створених нащадків симетричні щодо батьківських рішень. Також для фіксованого $\eta_c$ потомство має розподілення, пропорційне розподілу батьківських рішень.

Він має дві властивості:

– різниця між відповідними змінними рішеннями створеного потомства пропорційна різниці між відповідними змінними рішення батьківських рішень;

– скоріше буде обрано потомство, яке має змінні рішення, ближчі до батьківських рішень.

Поліноміальна мутація забезпечує ймовірність створення розчину поруч із батьківським вище, ніж ймовірність створення одного віддаленого від нього. Форма розподілу ймовірностей безпосередньо керується зовнішнім параметром $\eta_c$ і розподіл залишається незмінним протягом ітерацій.

У [46] пропонується використовувати модифіковану формулу обчислення відстані скупчення:

$$d^j = d^j + \frac{f_{n+1}^k - f_n^k}{f_{max}^k - f_{min}^k}$$

Така модифікація, по-перше, успадковує майже усі переваги початкового визначення, такі як відсутність параметрів, хороша характеристика розподілу тощо. По-друге, не додає алгоритму складності. Найбільшою перевагою є більш сильна здатність до сходження до фронту Парето.

ε-NSGAII – є ще одною модифікацією алгоритму, яка розширює NSGA-II шляхом додавання домінування $\varepsilon$, адаптивного розміру сукупності та само виключення, щоб мінімізувати потребу в калібруванні параметрів.

ε-домінування – це концепція, за допомогою якої користувач може задати точність, з якою він хоче отримати оптимальні за Парето рішення у багатокритеріальній задачі, по суті надаючи їм можливість надати відносну важливість кожній критерію.

Це досягається шляхом застосування сітки (розміром за вказаними користувачем ε значеннями) до простору пошуку. Більші значення ε приводять до кращої сітки (і, в кінцевому рахунку, менше рішень), тоді як менші значення $\varepsilon$ дають більш тонку сітку. Придатність кожного рішення потім відображається на бокс-фітнес на основі заданих значень $\varepsilon$. Потім сортування без домінування проводиться з використанням пристосованості коробки кожного рішення, а

рішення з однаковою придатністю коробки (тобто, рішення, що зустрічаються в одному блоці сітки), порівнюються, а рішення, які домінують у блоці сітки, усуваються. Це призводить до того, що в будь-якому одному блоці сітки існує не більше одного домінуючого рішення, що запобігає кластеризації рішень та сприяє більш рівномірному пошуку критеріального простору [47].

Недоліками NSGA-II можливо вважати наступне:

– ітераційний процес;

– легко створюються дублікати осіб, які викликають труднощі пошуку ізольованих точок;

– при збільшенні розмірності змінної конвергенція алгоритму NSGA-II значно зменшується [48].

Пропонується наступна модифікація методу NSGA-II, яка дозволяє знайти підмножину ефективних варіантів. Суть модифікації полягаю у наступному, обирається 20 альтернатив з загальної множини у якості однієї особи з популяції. Після того, як були обрані нові особини популяції алгоритм запускає ще одну ітерацію та отримується ново комбінація альтернатив, накопичені комбінації об'єднуються і формують підмножину.

Альтернативи у підмножині можуть повторятися, тому не усі оптимальні за Парето рішення попадуть до кінцевої підмножини варіантів. За умову зупинки алгоритму прийнято:  після трьох ітерацій та обрання особин поспіль, обрана особина не додала нові елементи до підмножини ефективних варіантів .

## 2.6 Метод випадкового пошуку

Методи випадкового пошуку використовують [49] розрахунок векторів критеріїв в випадкових точках. Відносно них проводиться відбір недомінуючих векторів. При цьому множина може модифікуватися в залежності від отриманих результатів або по попередньо заданою схемою.

Реалізація цього сімейства методів доволі легка, тому їх застосовують для вирішення різноманітних завдань проектування технічних систем. Збіжність

процес досягається при наближенні числа випадкових точок до нескінченності, хоча в таких методах оцінка якості апроксимації зазвичай не дається.

Випадковий пошук [50] виявляє себе як алгоритм з високим ступенем ефективності та у задах дискретно-неперервної оптимізації значно опереджає звичайні методи. При цьому не вимагає додаткового дослідження функції та використовується у задачах с великою кількістю параметрів.

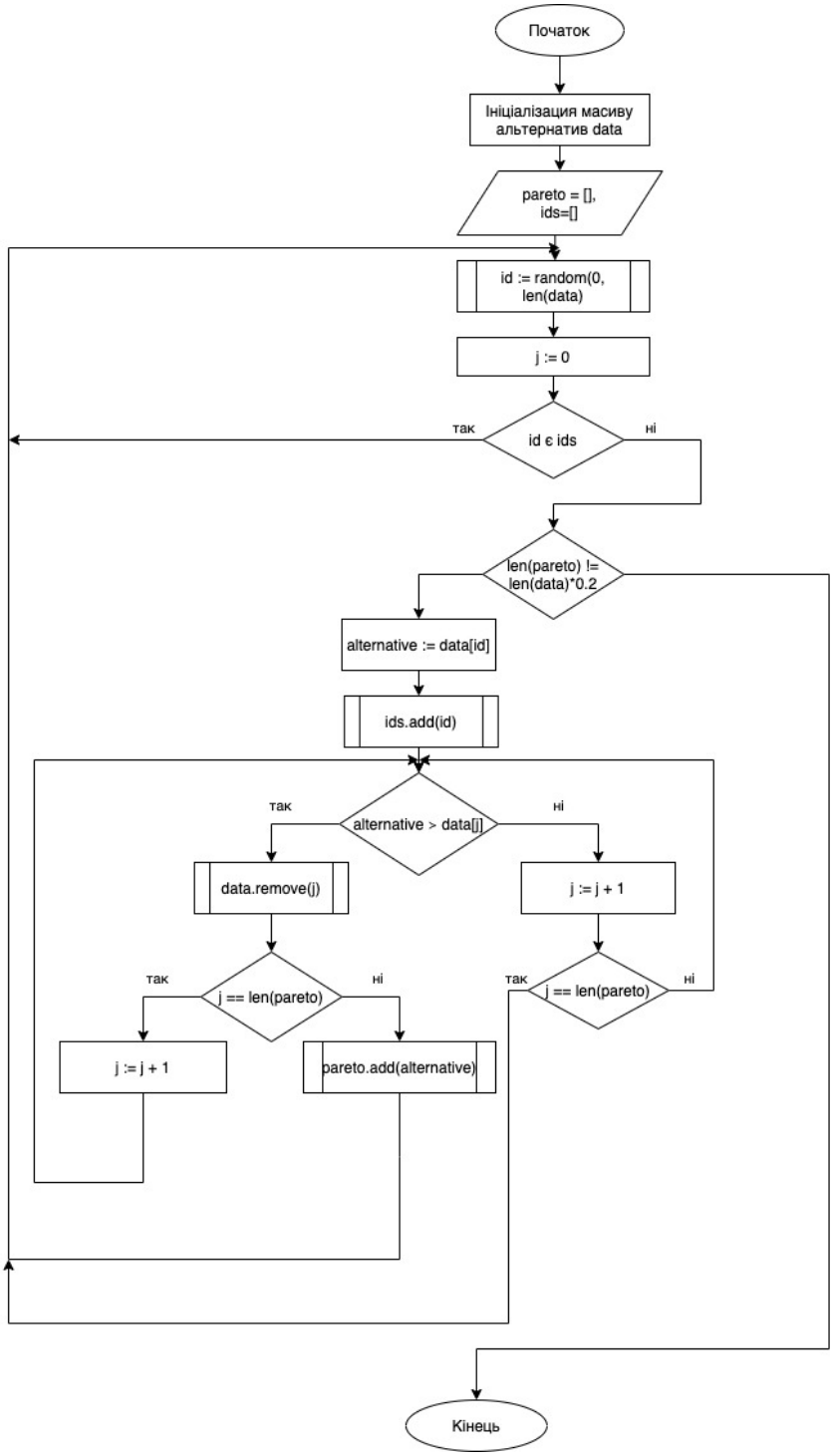


Рисунок 2.5 – Схема алгоритму випадкового пошуку

Проявленнями випадковість можуть бути [50]:

– закон розподілу довжини кроку спуску;

– моделювання закон розподілу напрямку спуску;

– координати вектору $x = \langle x_1, \dots, x_n \rangle$;

– розмір околиці пошуку.

Одним з варіантів використання методу випадкового пошуку у рамках множина Парето та підмножини ефективних варіантів поєднання методу прямого пошуку та стохастичного способу пошуку.

Перевагами методу прямого пошуку є напрямок, заснований на основі Парето оптимальності. Так розмір кроку визначається за допомогою нижньої та верхньої границі значень, а межа конвергенції, пов'язана з розміром кроку. Стохастичний метод пошуку знаходить глобальні оптимальні точки за допомогою поширення точок випадковим чином в області пошуку. Однак у цього методу є ще невелика ймовірність отримання глобальних оптимальних точок [51].

Окрім використання алгоритму випадкового пошуку самостійно, його різноманітні модифікації використовуються у еволюційних алгоритмах.

Оскільки вони працюють з сукупністю точок, багато Парето оптимальних рішень знаходять одночасно в одному циклі.

Один з таких – метод вибірки-пошуку-кластеризації. У цьому алгоритмі спочатку початкові точки відбираються з можливої області. Після того, як початкові точки отримані таким чином, їх локальним оптимізатором підштовхують до ефективної межі. Запропонований локальний оптимізатор – це алгоритм випадкового пошуку. Оскільки багато таких мінімальних рішень за одним критерієм можуть бути близькими один до одного, виконується кластеризація. Цей процес має ітеративний характер [52].

Було визначено, що деякі алгоритми випадкового пошуку показають кращі результати ніж NSGA-II, який вважається одним з найефективніших у своєму класі алгоритмів [53].

При використанні алгоритму випадкового пошуку потрібно приділяти багато уваги умовам зупинення алгоритму. Існує багато умов зупинки, деякі з них:

– повний перебір усіх елементів;

– зазначення проценту елементів, які мають узяти участь у роботі алгоритму до моменту його зупинки.

У даній роботі використовувався алгоритм, зображеній на рис. 2.5, при використані другої умови зупинки алгоритму, це було зумовлено тим, що при скупченості значень критеріїв при повному переборі елементів майже усі елементи залишалися у множені. Друга умова дещо скорочує кількість елементів, що потраплять до підмножини ефективних варіантів після останньої ітерації.

# 3 РОЗРОБКА ПРОГРАМНОГО ЗАСОБУ ТА АНАЛІЗ РЕЗУЛЬТАТІВ

## 3.1 Опис середовища розробки

В якості мови програмування використалась мова Python. Вона поєднує [54] у себе велику кількість бібліотек та пакетів з відкритим кодом. Вони охоплюють цілий спектр математичних областей, включаючи лінійну алгебру, числення, теорію чисел, криптографію, комутативну алгебру, теорію груп, комбінаторику, теорію графів та багато іншого. Наприклад, такі бібліотеки NumPy, SciPy, matplotlib та декілька інших бібліотек забезпечують числове обчислювальне середовище, схоже на Matlab, у той час як інші більш схожі на такі інструменти, як Mathematica, Maple або Magma.

Іншою перевагою використання мови програмування Python є те, що існує множина пакетів, які були написані іншими мовами, що забезпечує можливість використовувати особливості, які реалізовані на інших платформах та мовах. Наприклад, модуль CPython, написаний мовою програмування на C, також до інших реалізацій відносять IronPython (який працює з Microsoft .NET Framework), Jython (написаний на Java) та PyPy (написаний на Python). Реалізація Python CPython відрізняється тим, що зміни негайно обробляються інтерпретатором Python [55].

З іншого боку пакети, написані на Python легко інтегрувати у програмні засоби, написані на інших мовах. Це дозволяє зробити програму більш мобільною та використовувати її у ланці різноманітних проектів.

Однак, є ряд недоліків, на які потрібно звертати увагу при розробці складних алгоритмічних засобів:

– швидкість. Python працює більш повільніше, ніж C або C ++. Тому, що Python – мова високого рівня, на відміну від C або/чи C ++, які наближені до апаратних засобів;

– споживання пам'яті. Python для своєї праці потребує доволі великий запас пам’яті. Завдяки гнучкості типів даних споживання пам'яті у Python також високе.

Як середа розробки використовувався Jupyter Notebook, який є складовою частиною дистрибутива Anaconda, який окрім цього дозволяє управляти різними версіями мови, а також інтегрувати у проект стороні пакети. Середа розробки дозволяю виконувати по крокову кожну з команд.

Для відображення графічних даних використовувалась бібліотека з відкритим кодом Matplotlib, яка дозволяю будувати різноманітні графіки та відображують їх.

Середою виконання експериментів була машина з операційною системою Microsoft Windows 10 Pro з такими технічними характеристиками:

– RAM 4.00 GB;

– процесор AMD E1-6010 APU with AMD Radeon R2 Graphics.

## 3.2 Опис основних функцій

У рамках дослідження було розроблено програмний засіб «MOOptimize», який включає у себе такі функції:

– функція «Визначення множини Парето» (identify_pareto_set), яка у якості параметра приймає: values – масив значень, який містить значення по кожному з критерію для кожного рішення. Повертає масив самих рішень, які увійшли до множини та масив номерів цих альтернатив.

```
def identify_pareto_set(values):
    size = values.shape[0]
    pareto_ids = np.arange(size)
    pareto_front = np.ones(size, dtype=bool)
    for i in range(size):
        for j in range(size):
```

```
                if all(values[j] >= values[i]) and any(values[j] >
values[i]):
                    pareto_front[i] = 0
                    break
        return values[pareto_front], pareto_ids[pareto_front]
```

– функція «Кодування» (encode), яка у якості параметра приймає: value – значення, яке потрібно перетворити на бінарне; minLimit – граничне значення критерію (найменше); maxLimit – граничне значення критерію (найбільше). Повертає бінарне значення value.

```
    def encode(value, minLimit, maxLimit):
        hiPowFirst = hipow(minLimit,maxLimit)
        firstHiPow = hiPowFirst[0]
        highValue = hiPowFirst[1]
        ratio = float(format((highValue / np.power(2, firstHiPow)),
'.5f'))
        binValue = bin(math.floor(value / ratio * np.power(10,2)))
        binValue = ('00000000'+ binValue[2:])[-firstHiPow:]
        return binValue
```

– функція «Декодування» (decode), яка у якості параметра приймає: binary – значення, яке потрібно з бінарного перетворити на числове; minLimit – граничне значення критерію (найменше); maxLimit – граничне значення критерію (найбільше). Повертає бінарне значення value.

```
    def decode(binary, minLimit, maxLimit):
        hiPowFirst = hipow(minLimit,maxLimit)
        firstHiPow = hiPowFirst[0]
        highValue = hiPowFirst[1]
        ratio = float(format((highValue / np.power(2, firstHiPow)),
'.5f'))
        decimalNum = int(binary,2)
```

```
        converted = (decimalNum * ratio)/ np.power(10,2)
        return round(float((format(converted, '.5f'))), 3)
```

– функція «Формування хромосоми» (getChromosome), яка у якості параметрів приймає: values – масив значень, які будуть міститися у хромосомі; population_size – розмір популяції. Повертає хромосому у бінарному вигляді.

```
    def getChromosome(values, population_size):
        chromosomes = ['' for x in range(population_size)]
        for i in range(len(values)):
            objectives = values[i]
            for j in range(len(objectives[0])):
                limits = objectives[1]
                chromosomes[j] = chromosomes[j] +
str(encode(objectives[0][j], limits[0], limits[1]))
        return chromosomes
```

– функція «Швидке недомінуюче сортування» (fast_non_dominated_sort), яка у якості параметрів приймає: parent – значення фітнес функція батьківської популяції; children – значення фітнес функція популяції нащадків. Повертає масив Парето фронтів.

```
    def fast_non_dominated_sort(parent, children):
        S=[[] for i in range(0,len(parent))]
        front = [[]]
        n=[0 for i in range(0,len(parent))]
        rank = [0 for i in range(0, len(parent))]
        for p in range(0,len(parent)):
            S[p]=[]
            n[p]=0
            for q in range(0, len(parent)):
                if (parent[p] > parent[q] and children[p]
```

```
    > children[q]) or (parent[p] >= parent[q] and children[p] >
children[q]) or (parent[p] > parent[q] and children[p] >=
children[q]):
                    if q not in S[p]:
                        S[p].append(q)
                elif (parent[q] > parent[p] and children[q] >
children[p]) or (parent[q] >= parent[p] and children[q] >
children[p]) or (parent[q] > parent[p] and children[q] >=
children[p]):
                    n[p] = n[p] + 1
            if n[p]==0:
                rank[p] = 0
                if p not in front[0]:
                    front[0].append(p)
        i = 0
        while(front[i] != []):
            Q=[]
            for p in front[i]:
                for q in S[p]:
                    n[q] =n[q] - 1
                    if( n[q]==0):
                        rank[q]=i+1
                        if q not in Q:
                            Q.append(q)
            i = i+1
            front.append(Q)
        del front[len(front)-1]
        return front
```

– функція «Випадкового пошуку» (random_search), яка у якості параметра приймає: test_data – масив значень, який містить значення по кожному з критерію для кожного рішення. Повертає масив самих рішень, які увійшли до множини та масив номерів цих альтернатив.

```
    def random_search(test_data):
        rand_front = []
        new_ids= []
        size = len(test_data)
        first = randint(0, size-1)
        ids = []
        x = list(np.arange(size))
        while(len(ids) < len(test_data)*0.5):
            rand_int = randint(0, len(x)-1)
            ids.append(x[rand_int])
            rand_front.append(test_data[x[rand_int]])
            new_ids          =          compare2(len(rand_front),
np.array(rand_front))
            x.pop(rand_int)
        return                              np.array(ids)[new_ids],
np.array(rand_front)[new_ids]
```

– функція «Обчислення відстані щільності» (crowding_distance), яка у якості параметрів приймає: values – масив значень, який містить значення фітнес функцій для усіх рішень; size – розмір популяції; num_scores – кількість рішень для яких обчислюється фітнес функції. Повертає масив, який містить значення відстаней щільності для рішень, що містяться у values.

```
    def crowding_distance(values, size, num_scores):
        crowding_matrix = np.zeros((size, num_scores))
        for col in range(num_scores):
            crowding = np.zeros(size)
            crowding[0] = 1
            crowding[size - 1] = 1
            sorted_scores = np.sort(normed_scores[:, col])
            sorted_scores_index = np.argsort(
                values[:, col])
            crowding[1:size - 1] = \
                (sorted_scores[2:size] -
```

```
                sorted_scores[0:size - 2])
            crowding_matrix[:, col] = crowding[
np.argsort(sorted_scores_index)
]
    crowding_distances = np.sum(crowding_matrix, axis=1)
    return crowding_distances
```

### 3.3. Аналіз отриманих результатів

В ході дослідження було проведено серію експериментів з різними методами формування множини ефективних варіантів.

У якості вхідних даних можна використовувати, як дані з файлу, так і авто згенеровані дані. Так як потрібно провести серію експериментів з великою кількістю елементів, то використовувались згенеровані дані. Для деяких методів, також були потрібні вагові коефіцієнти критеріїв, в експериментах вважалося, що усі критерії рівноважні, тобто $\lambda_1 = \lambda_2 = \cdots = \lambda_i, \sum_{i=1}^{n} \lambda_i = 1$.

Для кожного методу було проведено 5 експериментів, які відрізнялися кількістю рішень у вхідних даних: 5000, 10000, 15000, 20000 та 25000. Кожна альтернатива оцінювалась за шістьма частковими критеріям, значення яких знаходиться у межах $[0, 1]$.

Таблиця 3.1 – Результати експериментів для множини з 10 000 варіантів

| Назва методу | Час роботи, с | Потужність підмножини компромісів |
|---|---|---|
| Попарних порівняння | 153,9 | 972 |
| Наближеної підмножини компромісів | 123,4 | 1274 |
| Лінійної згортки Карліна | 137,4 | 697 |
| Згортки Гермейєра | 144,1 | 837 |
| Еволюційний (алгоритм NSGA-II) | 92,4 | 952 |
| Еволюційний ( модифікованій алгоритм NSGA-II) | 89,2 | 967 |
| Випадковий пошук | 146,7 | 847 |

Для кожного з методів було встановлена динаміка часу визначення підмножин ефективних варіантів та побудовано її графічне відображення, яке включає також лінію тренду. В якості функції для побудови використовувався поліном третього ступеню.

Таблиця 3.2 – Час визначення підмножин ефективних варіантів методом парних порівнянь

| Кількість варіантів, $n$ | 5 000 | 10 000 | 15 000 | 20 000 | 25 000 |
|---|---|---|---|---|---|
| Час роботи, с | 53,8 | 153,9 | 290,3 | 532,4 | 634,2 |

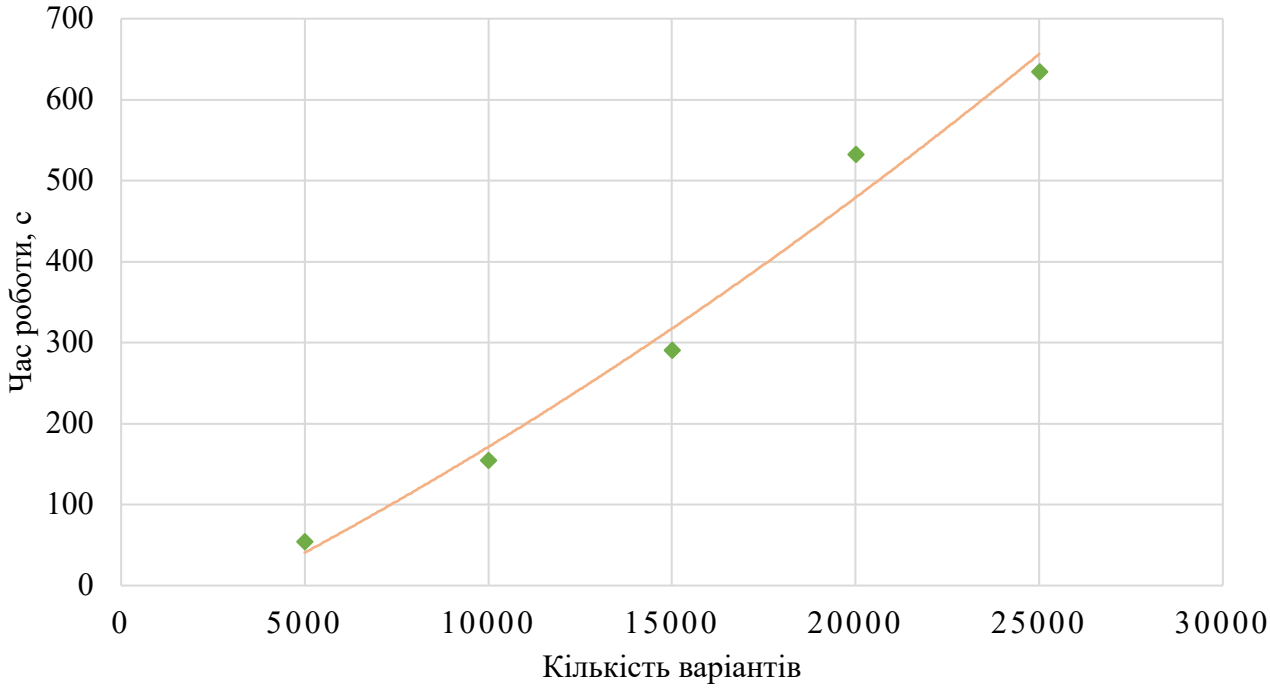


Рисунок 3.1 – Графічне відображення інформації з табл. 3.2

Часова складність визначення підмножин ефективних варіантів для методу попарних порівнянь складає:

$$T(n) = 3E - 07n^2 + 0{,}0214n - 74{,}3.$$

Таблиця 3.3 – Динаміка часу визначення підмножин ефективних варіантів при знаходженні наближеної області компромісів

| Кількість варіантів, $n$ | 5 000 | 10 000 | 15 000 | 20 000 | 25 000 |
|---|---|---|---|---|---|
| Час роботи, с | 25,1 | 123,4 | 218,6 | 437,4 | 580,3 |

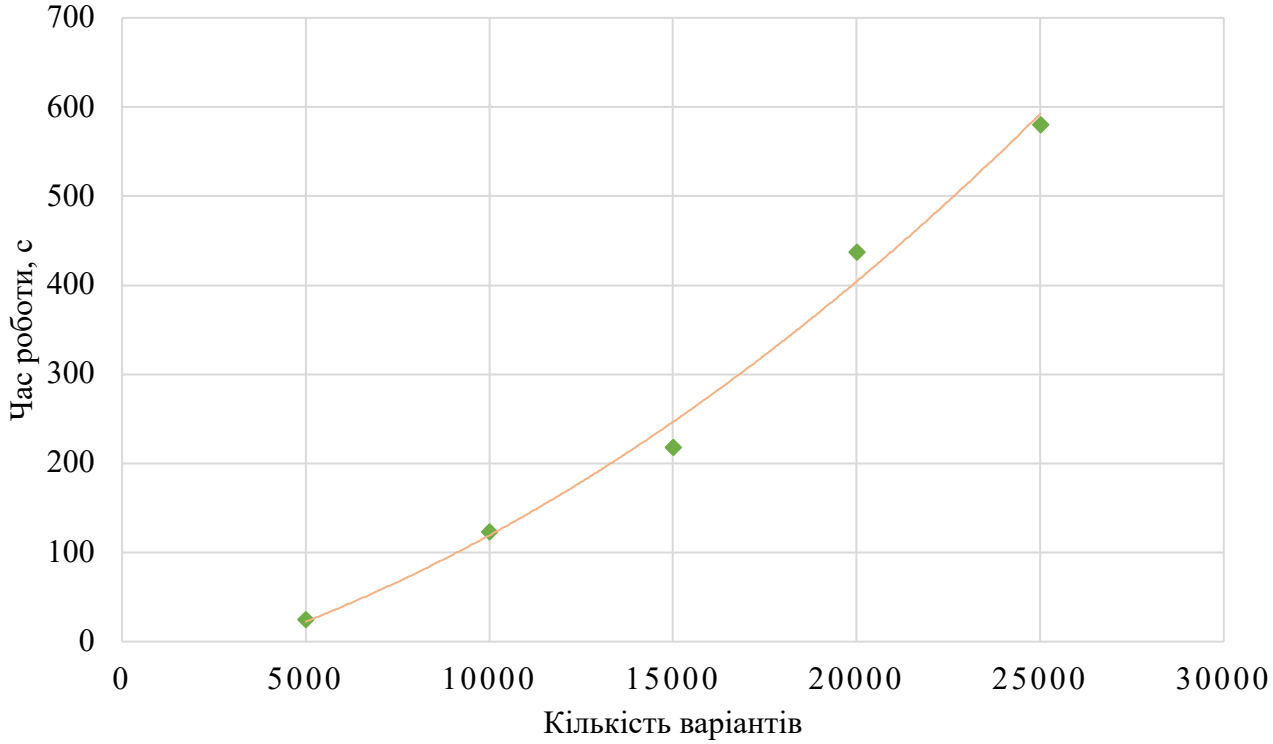


Рисунок 3.2 – Графічне відображення інформації з табл.3.3

Часова складність методу знаходженні наближеної множини компромісів складає:

$$T(n) = 6E - 07n^2 + 0{,}0102n - 43{,}96.$$

Таблиця 3.4 – Динаміка часу визначення підмножин ефективних варіантів для лінійної згортки (лема Карліна)

| Кількість варіантів $n$ | 5 000 | 10 000 | 15 000 | 20 000 | 25 000 |
|---|---|---|---|---|---|
| Час роботи, с | 43,3 | 137,4 | 260,7 | 459,3 | 525,5 |

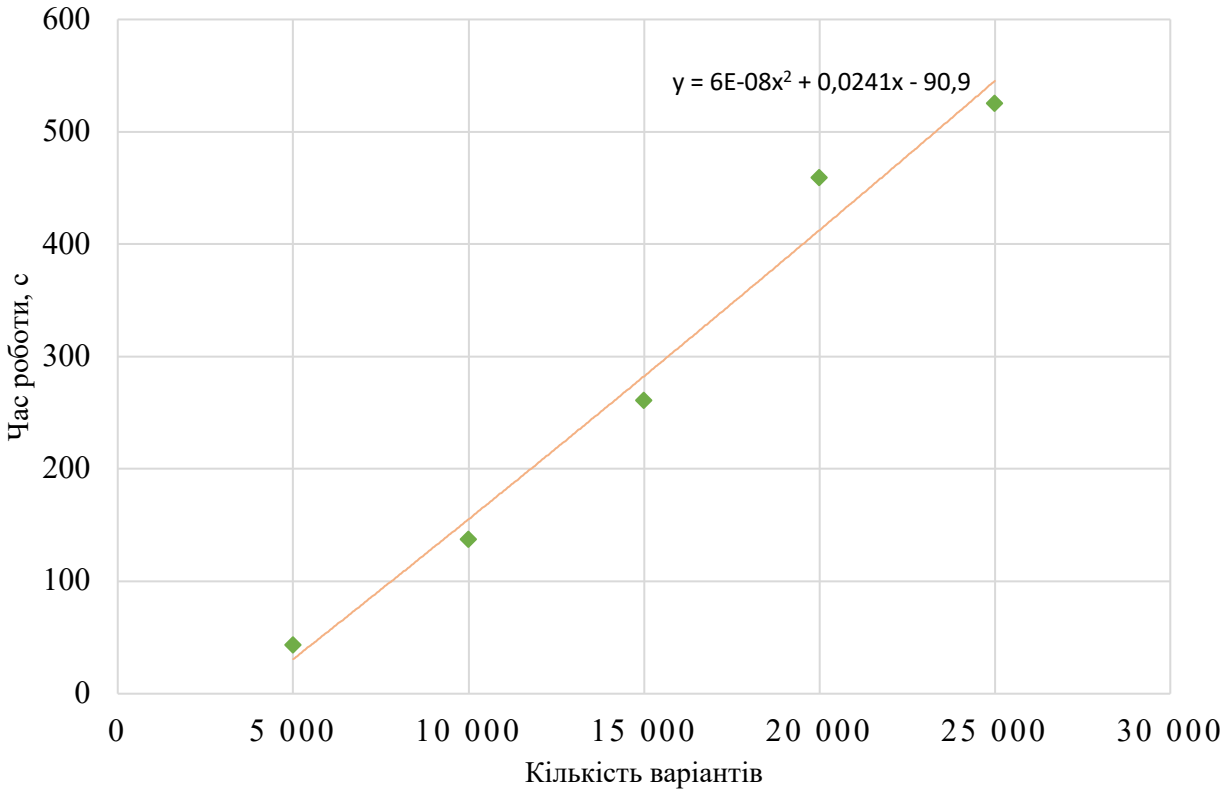


Рисунок 3.3 – Графічне відображення інформації з табл.3.4

Часова складність визначення підмножини ефективних варіантів при використанні лінійної згортки склала:

$$T(n) = 6E - 08n^2 + 0{,}0241n - 90{,}9.$$

Таблиця 3.5 – Динаміка часу визначення підмножин ефективних варіантів для згортки Гермейєра

| Кількість варіантів $n$ | 5 000 | 10 000 | 15 000 | 20 000 | 25 000 |
|---|---|---|---|---|---|
| Час роботи, с | 49 | 144,1 | 285,5 | 483,1 | 556,4 |

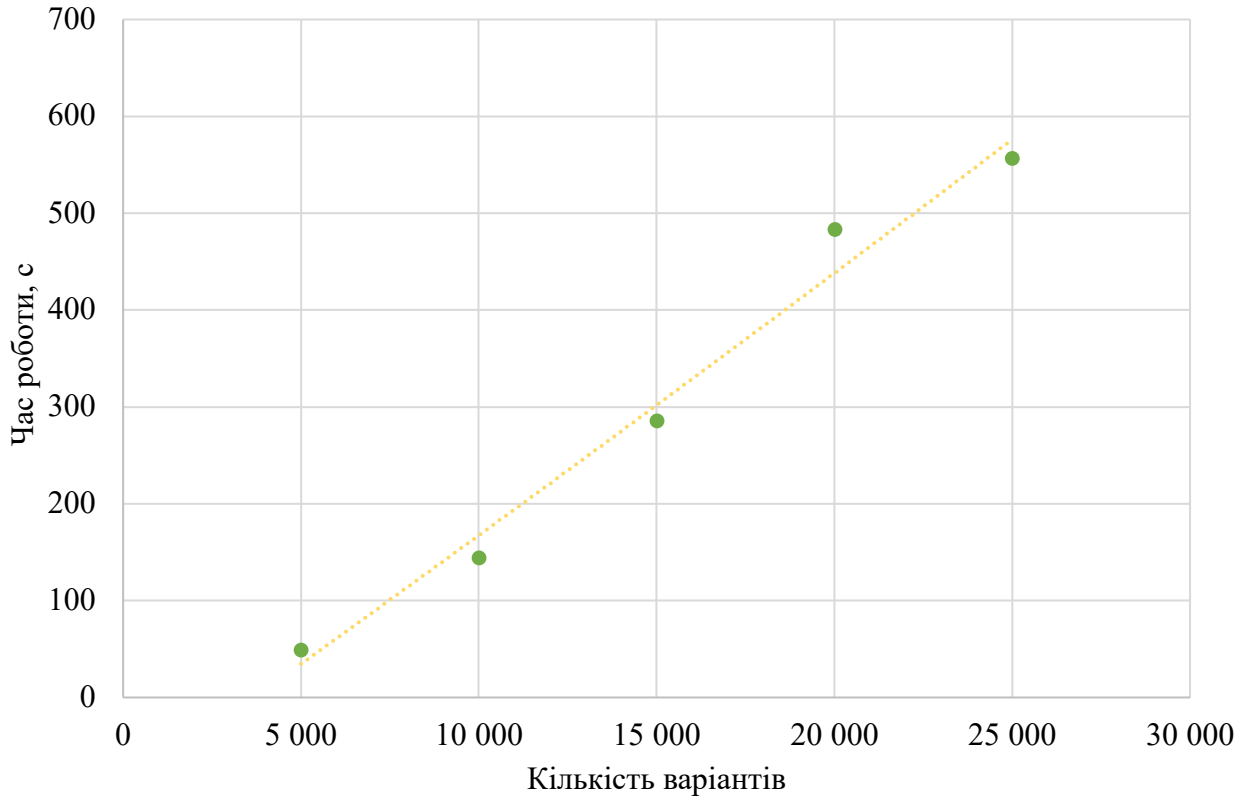


Рисунок 3.4 – Графічне відображення інформації з табл.3.5

Часова складність визначення підмножини ефективних варіантів при використанні згортки Гермейєра складає:

$$T(n) = 4E - 08n^2 + 0{,}026n - 96{,}22.$$

Таблиця 3.6 – Динаміка часу визначення підмножин ефективних варіантів для алгоритму NSGA-II

| Кількість варіантів $n$ | 5 000 | 10 000 | 15 000 | 20 000 | 25 000 |
|---|---|---|---|---|---|
| Час роботи, с | 28,5 | 92,4 | 189,1 | 343,8 | 455,1 |

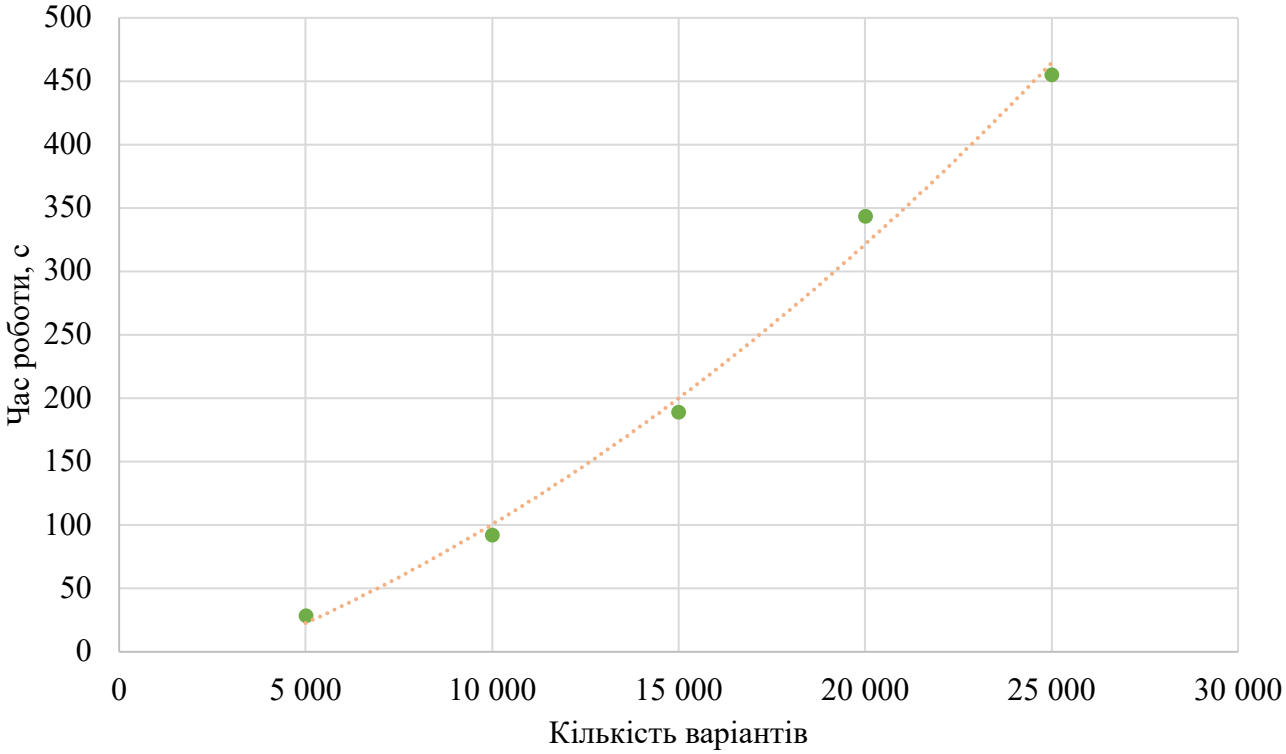


Рисунок 3.5 – Графічне відображення інформації з табл.3.6

Часова складність визначення підмножини ефективних варіантів при використані алгоритму NSGA-II склала:

$$T(n) = 4E - 07n^2 + 0{,}009n - 33{,}2.$$

Таблиця 3.7 – Динаміка часу визначення підмножин ефективних варіантів для модифікованого алгоритму NSGA-II

| Кількість варіантів $n$ | 5 000 | 10 000 | 15 000 | 20 000 | 25 000 |
|---|---|---|---|---|---|
| Час роботи, с | 25,4 | 89,2 | 177,6 | 323,4 | 445,4 |

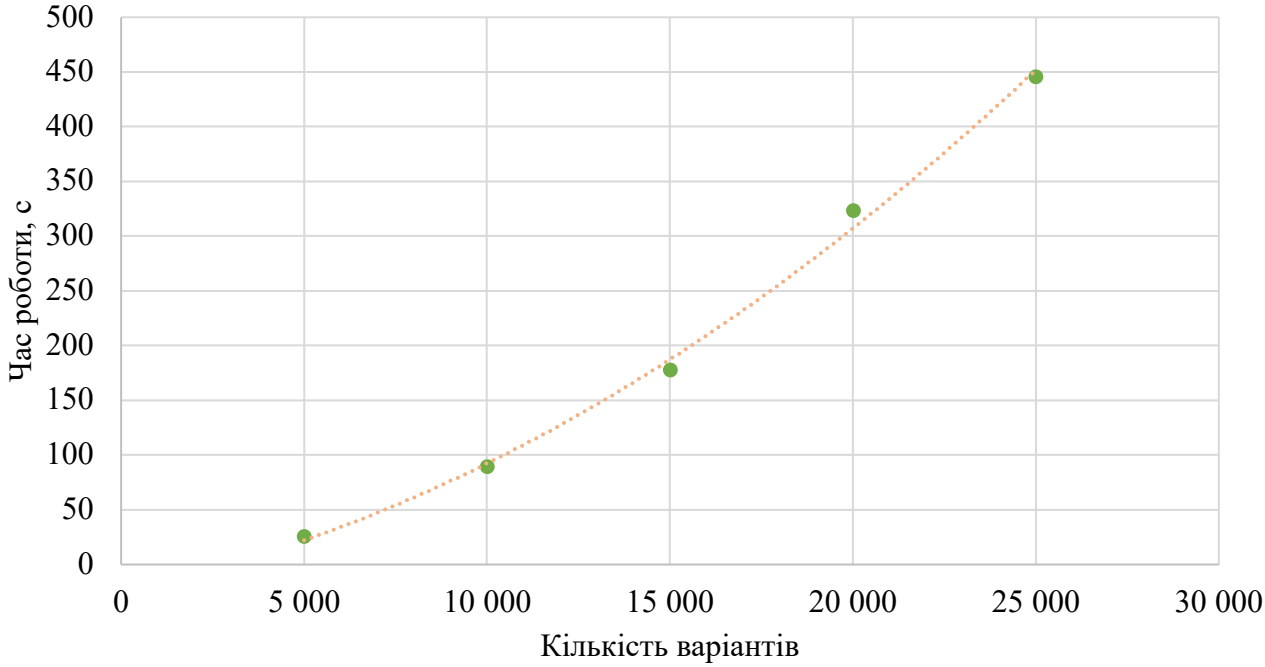


Рисунок 3.6 – Графічне відображення інформації з табл.3.7

Часову складність визначення підмножини ефективних варіантів при використані методу на основі модифікованого еволюційного алгоритму NSGA-II можна подати у вигляді:

$$T(n) = 5E - 07n^2 + 0{,}0066n - 23{,}16.$$

Таблиця 3.8 – Динаміка часу визначення підмножин ефективних варіантів для випадкового пошуку

| Кількість варіантів $n$ | 5 000 | 10 000 | 15 000 | 20 000 | 25 000 |
|---|---|---|---|---|---|
| Час роботи, с | 37,2 | 146,7 | 247,5 | 531,1 | 622,9 |

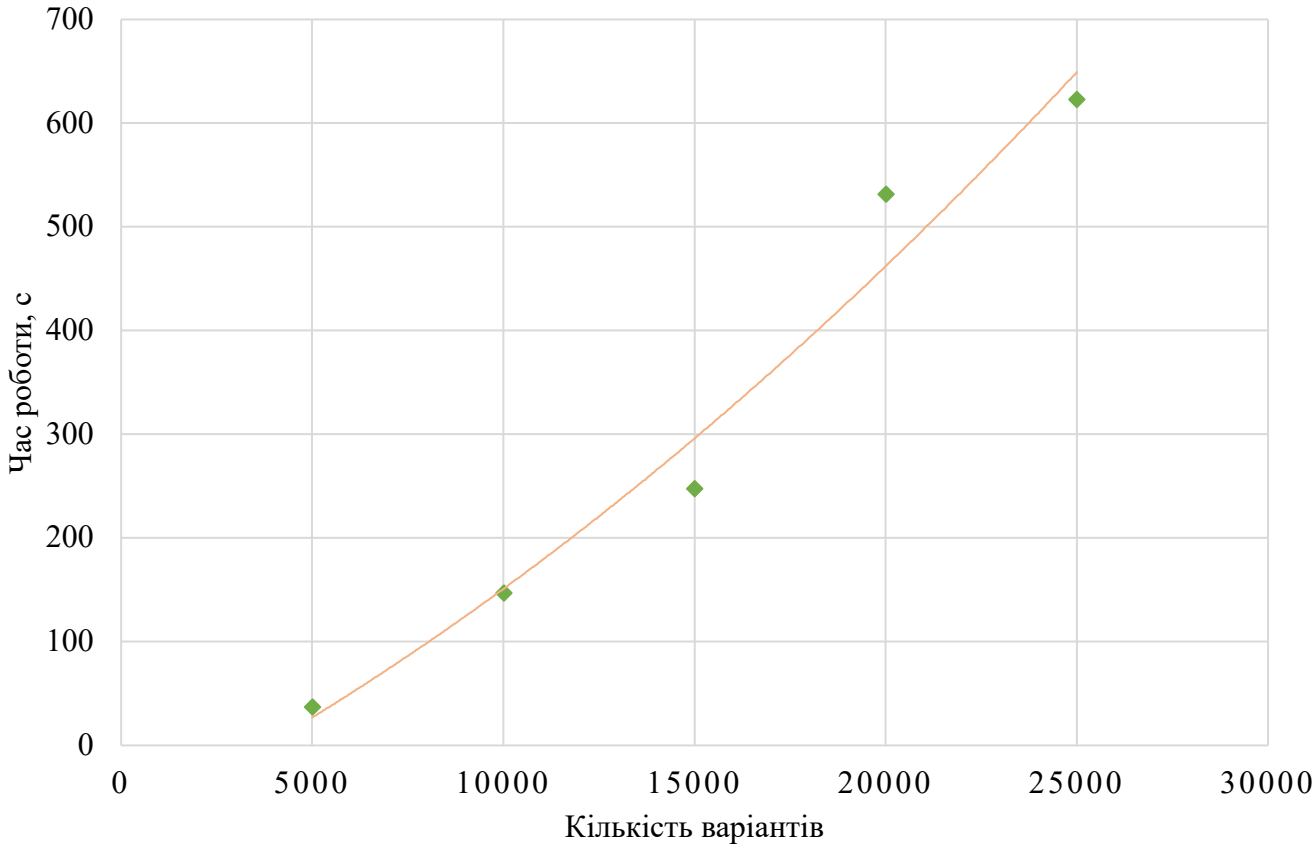


Рисунок 3.7 – Графічне відображення інформації з табл.3.8

Часову складність визначення підмножини ефективних варіантів для методу випадкового пошуку можна виразити наступним чином:

$$T(n) = 4E - 07n^2 + 0{,}0185n - 75{,}96.$$

Графічне відображення результатів дослідження для методів парних порівнянь, еволюційного методу на основі алгоритму NSGA-II та його запропонованої модифікації подано на рис. 3.8.

За результатами експериментального дослідження встановлено, що запропонована модифікація методу на основі алгоритму NSGA-II має приблизно на 3,46 % менший час розв’язання задачі та на 2 % точніше виділяє підмножину ефективних проектних рішень.

У порівняні з методом попарних порівнянь модифікація NSGA-II показує майже на 42 % менший час розв’язання задачі та лише на 0,5 % менш точніше виділяє підмножину ефективних проектних рішень.

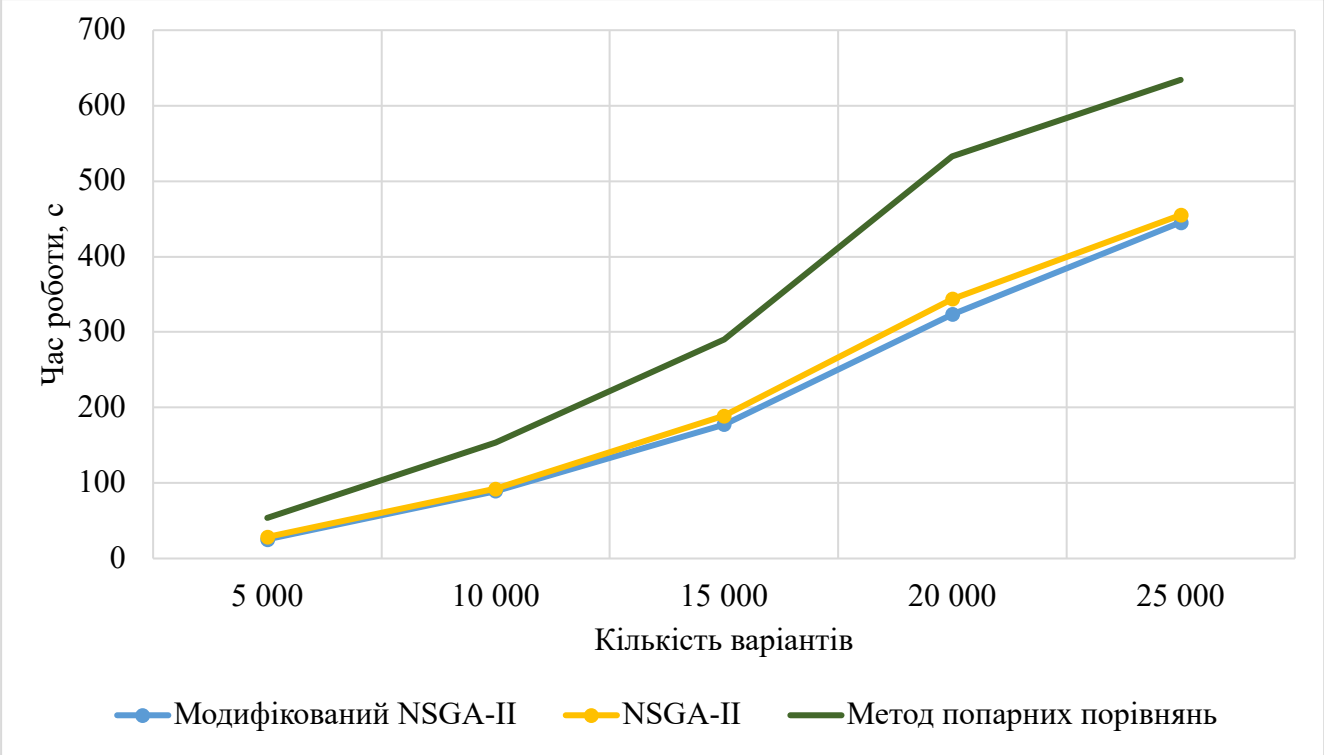


Рисунок 3.8 – Графічне відображення порівняння часової складності методів попарних порівнянь, еволюційного та його модифікації

Кожен з методів має своє переваги та недоліки, виходячи з вхідних даних та особливостей задачі багатокритеріальної оптимізації, такі як цільові функції, вагові коефіцієнти та вид вхідних даних.

Як видно з отриманих результаті при зростанні кількості вхідних елементів час на виконання алгоритмів зростає практично лінійно.

Методи Карліна та Гермейєра мають нижчу часову складність ніж метод попарного порівняння, метод знаходження наближеної підмножини компромісних варіантів та метод випадкового пошуку, але вони можуть не знаходити усі варіанти та саме ці методи залежать від суб'єктивних факторів, такі як вагові коефіцієнти, які надає особо, що приймає рішення.

Також як було зазначено лінійна згортка на основі метода Карліна не знаходить множину допустимих рішень, коли вхідна задача має вид опуклої множини. Цей недолік можна подолати за допомогою використання згортки Гермейєра. Ці два метода також пов'язує залежність від суб'єктивних ваг критеріїв, які мають бути визначені особою, яка приймає рішення або сгенеровані випадковим чином.

Алгоритм NSGA-II та його модифікація показали невелику часову складність. Але, його відрізняє велика алгоритмічна складність.

Метод випадкового пошуку зупиняє роботу, коли декілька альтернатив, у розмірі 20 відсотків від загальної кількості елементів множини будуть не вдалими. Цей метод має велику часову складність, тому що при визначенні випадковим чином альтернатив може використовуватися додатковий час. Значною перевагою над іншими розглянутими є невелика алгоритмічна складність. Не вдалими вважаються ті альтернативи, які не змінили підмножину ефективних рішень. Але недоліком методу випадкового пошуку може бути те, що не вся ефективні рішення попадуть до кінцевої множини, це залежить від умови зупинення алгоритму.

Метод попарних порівнянь має високу часову складність. Практично такі ж самі показники були отримані шляхом попереднього виділення наближеної множини компромісів. Але перевагою методу попарних порівнянь є те, що він

точно виділяє підмножину ефективних варіантів як на опуклих, так і на неопуклих множинах допустимих варіантів.

# ВИСНОВКИ

У роботі виконано дослідження методів визначення підмножин ефективних варіантів у системах підтримки прийняття проектних рішень. За результатами аналізу сучасного стану проблеми підтримки прийняття проектних рішень визначено як базові методи виділення підмножин ефективних рішень на основі попарних порівнянь, наближеної підмножини компромісів, Карліна, Гермейєра, еволюційний.

У роботі удосконалено еволюційний метод виділення підмножин ефективних проектних рішень, побудований на основі генетичного алгоритму NSGA-II, у частині зменшення його часової складності та підвищення точності. Розроблено алгоритми та програмне забезпечення розв’язання задачі. Це дозволило провести експериментальне дослідження розглянутих та удосконаленого методу та визначити оцінки їх точності та часової складності. За результатами дослідження встановлено переваги за точністю та часовою складністю розробленого варіанту еволюційного методу над його базовою версією.

Практичне використання отриманих результатів дозволить підвищувати ефективність систем підтримки прийняття багатокритеріальних проектних рішень за рахунок удосконалення процедур виділення підмножин ефективних варіантів.

Галузь застосування – відділи підприємств, організацій та компаній, що займаються проектуванням і плануванням розвитку електроенергетичних систем, корпоративного, місцевого чи регіонального рівня.

За результатами дослідження підготовлено доповіді на: 24-й Міжнародний молодіжний форум «Радіоелектроніка і молодь в XXI столітті» (м. Харків) [56]; звітну наукову конференцію викладачів, докторантів, аспірантів та студентів Прикарпатського національного університету ім. В. Стефаника (м. Івано-

Франківськ); міжнародну науково-практичну конференцію «Інформаційні технології в культурі, мистецтві, освіті, науці, економіці та бізнесі» (м. Київ).

# ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Шепель В.Н. Процедура принятия решения в экономико-математическом моделировании // Вестник Оренбургского государственного университета. 2011. № 13. С. 537-542.

2. Мехношин В.С. Системный анализ и принятие решения: учеб. пособие. Ульяновск : УВАУ ГА, 2006. 56 с.

3. Петровский А.Б. Теория принятия решений: учеб. пособие. Москва: Издательский центр «Академия», 2009. 400 с.

4. Multiple Attribute Decision Making Based on Cross-Evaluation with Uncertain Decision Parameters / Tao Ding, Liang Liang, Min Yang, Huaqing Wu // Mathematical Problems in Engeneering. 2016. p. 10.

5. A novel multi criteria decision making model for optimizing time–cost–quality trade-off problems in construction projects / Shahryar Monghasemi, Mohammad Reza Nikoo, Mohammad Ali Khaksar Fasaee, Jan Adamowski. Expert Systems with Applications. 2015. V. 42, №6. P. 3089-3104.

6. A rough set approach for determining weights of decision makers in group decision making / Qiang Yang, Ping-an Du, Yong Wang, Bin Liang // PLoS ONE. 2017. V. 12, №2.

7. Peter Keen. Decision support systems: a research perspective. Cambridge, Mass.: Center for Information Systems Research, Afred P. Sloan School of Management, 1980. P. 48.

8. Decision support system classification and its application in manufacturing sector: a review / Muhammad Syahid Hasana, Zuhriah Ebrahima, Wan Hasrulnizzam Wan Mahmoodb, Mohd Nizam Ab Rahmanc // Jurnal Teknologi. 2016. V. 79, №1.

9. Power, D. J. Decision Support Systems: A Historical Overview. Berlin: Springer Berlin Heidelberg, 2008. P. 121–140.

10. Raheja, V., S. Mahajan. Decision Support System, Its Components, Model and Types of Managerial Decisions // International Journal of Innovative Research and Studies. 2013. V. 2, № 12. P. 413-418.

11. Beraldi, P., A. Violi, F. De Simone. A Decision Support System For Strategic Asset Allocation // Decision Support Systems. 2011. V. 51, № 3, p. 549-561.

12. Marakas G. M. Decision support systems in the twenty-first century. Upper Saddle River. N. J.: Prentice Hall, 1999. 528 p.

13. Попов А.Л. Системы поддержки принятия решений: учеб. пособие. Екатеринбург: Урал. гос. ун-т, 2008. 80 с.

14. Петровский А.Б. Теория принятия решений: учеб. пособие. Москва: Издательский центр «Академия», 2009. 400 с.

15. Блюмин С.Л., Шуйкова И.А. Модели и методы принятия решений в условиях неопределенности. Липецк: ЛЭГИ, 2001. 138 с.

16. Ларичев О.И., Мошкович Е.М. Качественные методы принятия решений. Вербальный анализ решений. М.: Наука, 1996. 208 с.

17. Борисов А.Н., Вилюмс Э.Р., Сукур Л.Я. Диалоговые системы принятия решений на базе мини-ЭВМ. 1986. 196 с.

18. Saaty T.L. Decision making with the analytic hierarchy process // Int. J. Services Sciences. 2008. V. 1, № 1. P. 83–98.

19. Ricardo Viana Vargas. Using the analytic hierarchy process (AHP) to select and prioritize projects in a portfolio. 2010. P. 2-5.

20. Vasiliki Balioti, Christos Tzimopoulos, Christos Evangelides. Multi-Criteria Decision Making Using TOPSIS Method Under Fuzzy Environment. Application in Spillway Selection // Procedings. 2018. № 2. P. 2-3.

21. Ewa Roszkowska. Multi-criteria decision making models by applying the TOPSIS method to crisp and interval data // Multiple Criteria Decision Making. 2011. Vol. 6. P. 200-230.

22. Dede Wira Trise Putra, Adrian Agustian Punggara. Comparison Analysis of Simple Additive Weighting (SAW) and Weigthed Product (WP) In Decision Support Systems // MATEC Web of Conferences. 2018. № 215. P. 1-2.

23. Valentinas Podvezko. The Comparative Analysis of MCDA Methods SAW and COPRAS // Inzinerine Ekonomika-Engineering Economics. 2011. Vol. 22, №2. P. 134-146.

24. Бескоровайный В. В., Замирец О. Н., Настенко С. В. Методы определения подмножества эффективных решений при проектировании крупномасштабных объектов. Технология приборостроения. 2016. № 6. С. 7–10.

25. Gabriel Dimitriua, Razvan Ştefanescub, Ionel M. Navon. Comparative numerical analysis using reduced-order modelling strategies for nonlinear large-scale systems // Journal of Computational and Applied Mathematics. 2017. № 310. P. 32–43.

26. Бескоровайный В. В., Красько А.Ф. Автоматизация процессов выбора эффективных решений при автоматизированном проектировании систем управления и автоматики. Вестник Херсонского национального технического университета. 2007. №4 (27). С. 208–212.

27. E. Abel, L. Mikhailov, J. Keane. Group aggregation of pairwise comparisons using multi-objective optimization Inf. Sci., 322 (2015), pp. 257-275

28. Ramik, Jaroslav. Ranking Alternatives by Pairwise Comparisons Matrix and Priority Vector // Scientific Annals of Economics and Business. 2017.

29. Ногин В.Д. Принятие решений в многокритериальной среде: количественный подход. М.: ФИЗМАТЛИТ, 2002. С. 21-26.

30. Бескоровайный В.В., Настенко С.В. Формирование подмножества эффективных вариантов при реинжиниринге структур крупномасштабных объектов // Информационные системы и технологии: материалы 5-й Международ. науч.-техн. конф., Харьков, 12-17 сентября 2016 г.: тезисы докладов. Х.: ДРУКАРНЯ МАДРИД. 2016. С. 21-22

31. Петров Е.Г., Новожилова М.В., Гребеннік І.В. Методи та засоби прийняття рішень у соціально-економічних системах: навч. посібн. К. : Техніка, 2004. С. 15-16.

32. А.Н. Костенко. Определение области компромиссов при решении задач оптимизации в многокритериальной постановке // Системи обробки інформації. 2000. №3. С.54.

33. Trivedi V., Varshney P., Ramteke M. A simplified multi-objective particle swarm optimization algorithm. Swarm Intell. 2020. № 14. P. 83–116.

34. Fu, G., Wang, C., Zhang, D. A Multiobjective Particle Swarm Optimization Algorithm Based on Multipopulation Coevolution for Weapon-Target Assignment. // Math. Problems Eng. 2019. №4. P. 123-127.

35. Ногин В.Д. Линейная свертка в многокритериальной оптимизации // Искусственный интеллект и принятие решений. 2014. № 4. С. 73-82.

36. Аристова Е.М. Установление взаимосвязи между методами аддитивной свертки и метрики // Вестник Дагестанского государственного технического университета. 2017. № 44. С. 107-117.

37. Лотов А.В., Поспелова И.И. Многокритериальные задачи принятия решений: Учебное пособие. М. : МАКС Пресс, 2008. 197 с

38. Ногин В.Д. Проблема сужения множества Парето: подходы к решению // Искусственный интеллект и принятие решений. 2008. №1. С. 98-112.

39. Романова И.К. Об одном подходе к определению весовых коэффициентов метода пространства состояний // Наука и образование. МГТУ им. Н.Э. Баумана. 2015. № 4. С. 105-129.

40. Lei Xiujuan, Shi Zhongke. Overview of multi-objective optimization methods // Journal of Systems Engineering and Electronics. 2004. V. 15. №2. P. 142-146.

41. Городецкий С.Ю., Гришагин В.А. Учебный курс «Модели и методы конечномерной оптимизации». Нижний Новгород, 2003. 257 с.

42. Ногин В.Д. Сужение множества Парето на основе аксиоматического подхода с применением некоторых метрик // Журнал вычислительной математики и математической физики. Т. 57. №4. 2017. С. 645-653.

43. Giagkiozis, I. and Fleming, P.J. Methods for multi-objective optimization: An analysis // Information Sciences. 2015. № 3. P. 338-350.

44. Deb, K., Pratap, A., Agarwal, S. A fast and elitist multiobjective genetic algorithm: NSGA-II // IEEE Transactions on Evolutionary Computation. 2002. №6(2). P. 182–197.

45. Murugan, P., Kannan, S., Baskar, S. NSGA-II algorithm for multi-objective generation expansion

46. X. Chu and X. Yu. Improved Crowding Distance for NSGA-II // Information Technologies. 2018. V. 18. P. 126-130.

47. Kollat J.B., Reed P.M. The Value of Online Adaptive Search: A Performance Comparison of NSGAII, ε-NSGAII and εMOEA // Lecture Notes in Computer Science. 2005. V. 34. P. 211-217.

48. Fang, X., Wang, W., He, L. Research on Improved NSGA-II Algorithm and Its Application in Emergency Management // Mathematical Problems in Engineering 2018. №7 P. 1–13.

49. Берёзкин В.Е., Каменев Г.К., Лотов А.В. Гибридные адаптивные методы аппроксимации невыпуклой многомерной границы Парето // Ж. вычисл. матем. и матем. физ. 2006. Т. 46, № 11, С. 2009–2023.

50. Кушербаева В.Т., Сушков Ю.А. Статистическое нсследовапие случайного поиска // Стохастическая оптимизация в информатике / Под ред. О. Н. Граничина. 2007. С. 21-36.

51. Na S., Karr D. Development of Pareto strategy multi-objective function method for the optimum design of ship structures // International Journal of Naval Architecture and Ocean Engineering. 2016. № 8. P. 602-614.

52. Shukla P., Deb K. On finding multiple Pareto-optimal solutions using classical and evolutionary generating methods // Eur. J. Oper. Res. 2007. V.. 181, № 3. P. 1630–1652.

53. Mostaghim S, Schmeck H. Distance based ranking in many-objective particle swarm optimization // Lecture notes in computer science. 2008. V. 5199, №12. P. 753–762.

54. Millman K. J., Aivazis M. Python for Scientists and Engineers // Computing in Science & Engineering. 2011. V. 13, № 2. P. 9-12.

55. John B., Schneider Shira, Lynn Broschat. Algorithmic Problem Solving with Python // Information Technologies. 2019. №5. P.7-8.

56. Вакуленко В.К. Визначення підмножин ефективних варіантів при прийнятті багатокритеріальних проектних рішень // Радіоелектроніка та молодь у XXI столітті: матеріали 24-го Міжнародного молодіжного форуму. 2020. Т. 6. С. 286-287.