МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ

ЦЕНТР ЗАОЧНОЇ, ДИСТАНЦІЙНОЇ ТА ВЕЧІРНЬОЇ ФОРМ НАВЧАННЯ

КАФЕДРА КОМП'ЮТЕРНИХ НАУК

СЕКЦІЯ ІКТ

КВАЛІФІКАЦІЙНА МАГІСТЕРСЬКА РОБОТА

на тему:

«Порівняльний аналіз алгоритмів чисельного розв'язання задач нелінійного програмування. Алгоритм багатокритеріальної оптимізації»

Завідувач

випускаючої кафедри **Довбиш А. С.**

Керівник роботи **Шаповалов С. П.**

Студент групи ІНмз – 91с **Михайлов В. Т.**

СУМИ 2020

Сумський державний університет

(назва вузу)

Факультет *ЦЗДВФН* Кафедра *Комп'ютерних наук*

Спеціальність *«Інформатика»*

Затверджую:

зав.кафедрою _____

_____

"_____" _____ 20___р.

ЗАВДАННЯ

НА ДИПЛОМНИЙ ПРОЕКТ (РОБОТУ) СТУДЕНТОВІ

Михайлову Владиславу Тимофійовичу

(прізвище, ім'я, по батькові)

1. Тема проекту (роботи) *«Порівняльний аналіз алгоритмів чисельного розв'язання задач нелінійного програмування. Алгоритм багатокритеріальної оптимізації»*

затверджую наказом по інституту від "____" _____ 20 ___ р. № _____

2. Термін здачі студентом закінченого проекту (роботи) _____

3. Вхідні данні до проекту (роботи) _____
_____
_____

4. Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити) *1) Аналіз проблеми. Постановка задачі дослідження. 2) Інформаційний огляд. 3) Математична модель та вибір методу рішення 4) Розробка інформаційного та програмного забезпечення системи*

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

6. Консультанти до проекту (роботи), із значенням розділів проекту, що стосується їх

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання _____

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів дипломного проекту (роботи)	Термін виконання проекту (роботи)	Примітка
1.	*Аналіз проблеми. Постановка задачі дослідження*		
2.	*Інформаційний огляд*		
3.	*Математична модель та вибір методу рішення*		
4.	*Розробка інформаційного та програмного забезпечення системи*		
5.	*Оформлення пояснювальної записки до дипломної роботи*		

Студент – дипломник _____

(підпис)

Керівник проекту _____

(підпис)

РЕФЕРАТ

Записка: 34 стор., 7 рис., 2 табл., 1 додаток, 8 джерел інформації.

Об’єкт дослідження — задачі багатокритеріальної оптимізації.

Мета роботи — комп’ютерна реалізація алгоритму багатокритеріальної оптимізації для рішення задач нелінійного програмування.

Методи дослідження — математичне моделювання, алгоритм багатокритеріальної оптимізації, комп’ютерна реалізація алгоритмів на ЕОМ.

Результати — впроваджено алгоритм багатокритеріальної оптимізації для рішення задач нелінійного програмування. Проведено огляд відомих рішень, обрано алгоритм для розв’язання трикритеріальних задач програмування. Проведені тестові випробування за допомогою алгоритмічної мови програмування Java.

НЕЛІНІЙНЕ ПРОГРАМУВАННЯ, КОМПЮТЕРНА РЕАЛІЗАЦІЯ,
БАГАТОКРИТЕРІАЛЬНІ ЗАДАЧІ, ПОСТАНОВКА ЗАДАЧІ,
ТЕСТОВІ РОЗРАХУНКИ

ЗМІСТ

ВСТУП

Серед всієї множини задач вибору (оптимізації), що виникають при дослідженні складних систем, завжди актуальною є задача пошуку оптимального рішення за кількома критеріями. У будь-який технічній системі, або системі організаційного типу, є такі характеристики, кількісні значення яких бажано максимізувати, а є й такі, які бажано кількісно зменшити за всіма мірками [1-4].

Під багатокритеріальною задачею найчастіше розуміють не власне вербальний опис задачі, а її модель, а саме: багатокритеріальна задача – це математична модель прийняття оптимального рішення за декількома критеріями. Ці критерії можуть відображати оцінки різних якостей об'єкта або процесу, з приводу яких приймається рішення.

Але між окремими характеристиками, як правило, існує взаємна залежність та діє ряд обмежень. Внаслідок цього виявляється, що поза деякою областю, що називається областю узгодження, збільшення одних характеристик тягне за собою зменшення інших, причому тих, чисельні значення яких бажано також збільшувати і навпаки. Область, де характеристики взаємообумовлені, називається областю компромісу.

Хоча ефективний розв’язок зазвичай буває далеко не єдиним, але все ж таки множина ефективних альтернатив значно вужча, ніж вихідна множина всіх розв’язків. З огляду на це, побудова множини ефективних розв’язків (або їх оцінок) є першим етапом здійснення великої кількості процедур і методів багатокритерійної оптимізації [5-7].

Для задач оптимізації фундаментальними є поняття математичної моделі об'єкта і ті дані, якими дослідник оперує для побудови моделі. Спектр широкий - від повного знання до повної невизначеності. Між цими інформаційними полюсами знаходиться імовірнісний рівень невизначеності. Наявність достатньої інформації про механізми фізичних, хімічних, інформаційних, економічних і інших процесів, що відбуваються в об'єкті, дозволяє скласти детерміновану модель у вигляді диференціальних, алгебраїчних і інших рівнянь.

Аналітичне дослідження щодо простих детермінованих математичних моделей є предметом класичної теорії оптимізації.

Багатоцільова оптимізація застосовується у багатьох галузях науки, включаючи машинобудування, економіку та логістику, де потрібно приймати оптимальні рішення за наявності компромісів між двома або більше суперечливими цілями. Мінімізація витрат при максимальному комфорті під час покупки автомобіля та максимізація продуктивності при мінімізації споживання палива та викидів забруднюючих речовин автомобіля є прикладами багатоцільових задач оптимізації, що включають дві та три цілі відповідно. У практичних задачах може бути більше трьох цілей.

1 АНАЛІТИЧНИЙ ОГЛЯД ПРОБЛЕМИ

1.1 Алгоритми та методи розв'язку багатокритеріальних задач

Завдання багатокритеріальної оптимізації - це завдання з декількома критеріями, які з різних сторін характеризують різні рішення. Найчастіше заздалегідь виділено напрямок поліпшення кожного критерію, наприклад його збільшення. Але одночасне збільшення всіх критеріїв практично завжди неможливо.

Наприклад, програми розвитку, що розроблюються в процесі управління підприємствами, основані на пошуку найбільш ефективного набору проектів, тобто варіантів удосконалення виробничих і організаційних процесів. Широкий спектр показників, що впливає на результативність діяльності, приводить до необхідності пошуку компромісу між різними критеріями, що веде до багатокритеріальних задач вибору.

Існуючі на сьогодення алгоритми та методи рішення завдань багатокритеріальної оптимізації умовно можна розподілити на дві основні групи.

До першої з них відносять такі, що зводять завдання багатокритеріальної оптимізації до задачі однокритеріальної, шляхом звертання всіх поставлених критеріїв до одного суперкритерію, з подальшим застосуванням одного з вибраних методів рішення однокритеріальної оптимізації. Для такого шляху розроблено великий арсенал видів згорток, за допомогою яких розвязують складні завдання великої розмірності [3-5]. Але в теорії прийняття рішень прийнято вважати такі методи неефективними.

Всі інші методи, що залишилися, автоматично переходять до другої групи:

- методи та алгоритми послідовних поступок;
- методи аналізу ієрархій;
- алгоритми наближення до ідеального рішення;
- генетичні алгоритми;

Розв'язання проблем багатокритеріальної алгоритмами та методами першої групи.

Найбільш вживаними підходами до формування узагальненого критерію є такі згортки критеріїв:

$$F_1(x) = \sum_{i=1}^{m} \rho_i k_i(x),$$

А) лінійні (адитивні) ; (1.1)

де ki - критерій оптимізації, ρi - вага i-го критерію, i - номер критерію (I = 1, m);

$$F_2(x) = \prod_{i=1}^{m} (k_i(x))^{\rho_i} ;$$

Б) мультиплікативні (1.2)

В) комбіновані, наприклад, представимо у вигляді функціоналу F3 (F1, F2), залежить від параметрів:

$$F_3(F_1,F_2) = \sum_{j=1}^{2} \mu_j F_j(x), \quad \text{где } 0 \le \rho_i, \mu_j \le 1, \sum_{i=1}^{m} \rho_i = 1, \sum_{j=1}^{n} \mu_j = 1; \tag{1.3}$$

тут μ j - вага j-го комплексного критерію (j = 1, n);

Г) максиміний (мінімаксний) критерії:

$$F_4(x) = min\ max\ k_i(x) \text{ или } F_4(x) = max\ min\ k_i(x). \tag{1.4}$$

Перевагу віддають адитивному критерію, якщо істотне значення для даної задачі мають абсолютні значення критеріїв для обраного набору параметрів. Мультиплікативний критерій доцільно вибирати, якщо істотну роль відіграє зміна абсолютних значень окремих критеріїв при варіації шуканого параметра. Що стосується виконання завдання досягнення рівності нормованих значень суперечливих приватних критеріїв вибирають максимінний або мінімаксний критерій.

Вибір критеріїв ефективності є найбільш відповідальним і складним етапом при постановці завдання, оскільки цілі, до яких прагне модельований об'єкт, часто багатогранні і суперечливі.

Для усунення зазначених недоліків можна використовувати метод, який заснований на формуванні комплексних оцінок критеріїв, отриманих шляхом побудови ієрархічної структури дерева цільових критеріїв, і пошуку оптимального рішення в просторі отриманих комплексних критеріїв (див. рис. 1.1).

Ідея даного підходу, перш за все, передбачає виявлення в процесі діагностики підприємства показників ефективності, які будуть використовуватися для оцінки ефективності його діяльності. Паралельно необхідно оцінити, наскільки

цінно для підприємства досягнення критерієм того чи іншого значення. На підставі отриманої інформації слід побудувати ієрархічну структуру критеріїв і розрахувати сукупну оцінку критеріїв для кожної "гілки" дерева, з огляду на його структуру на нижніх рівнях.

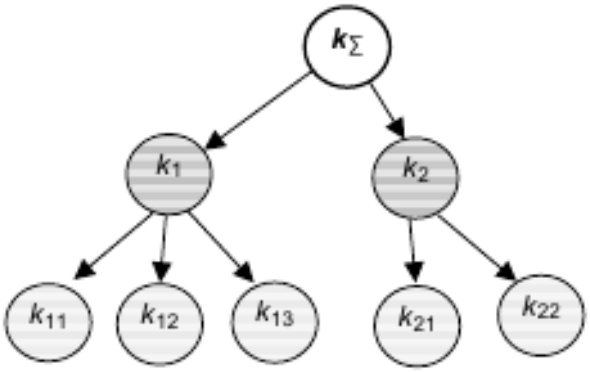

Рисунок 1. 1 – Ієрархічна структура критеріїв ефективності

Припустимо, кожен варіант розвитку підприємства характеризується п'ятьма показниками. Умовно назвемо їх: дохід від впровадження (визначає кількість продажів, тобто частку охопленого конкурентного ринку); витрати на реалізацію (характеризує зміни собівартості продукції); економічна ефективність (питомий дохід або питомі витрати); невизначеність (кількість інформації про наслідки впровадження) та безпеку (технічна, технологічна, інформаційна тощо). Після угруповання критеріїв розташуємо їх за рівнями ієрархії. На рис. 1.1 критерії позначені як $К_{11}$, $К_{12}$, $К_{13}$, $К_{21}$, $К_{22}$.

Важливим ефектом використання "дерева" критеріїв є поділ відповідальності експертів щодо запропонованих напрямками і рівнями. У цьому випадку кожен фахівець буде відповідати за деяку конкретну область і формувати проміжну оцінку, що її характеризує. Таким чином, з урахуванням рівнів будуть сформовані вагові коефіцієнти ρ_{ij} для кожного з критеріїв

($i = 1, m_j$, де m_j - кількість критеріїв на "гілці").

Для отримання комплексних критеріїв k1 (прибутковість) і k2 (рівень ризику) будемо використовувати згортки з класу лінійних виду

$$k_j = \sum_{i=1}^{m_j} \rho_{ij} k_{ij} \,. \qquad (1.5)$$

Такий клас є найбільш доступним для огляду, проте, в свою чергу, володіє недоліком: можливі ситуації, коли всі ефективні рішення не будуть враховані. Показвний на рис. 1. 2, а план S2 буде втрачено, оскільки при будь-яких значеннях вагових коефіцієнтів ρ 1 і ρ 2 будуть обрані плани S1 і S3. Для вирішення виниклої проблеми необхідно вдатися до нелінійним перетворенням:

$$k_1' = f(k_1), \quad k_2' = f(k_2). \tag{1.6}$$

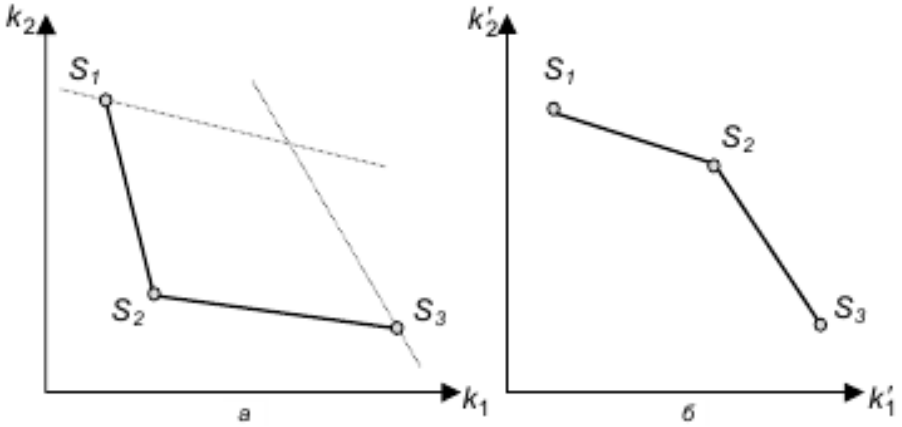

Рисунок 1.2 – Нелінійні перетворення критеріїв

У разі, як на рис. 2, б, можна підібрати значення ваг так, що жоден з варіантів не буде упущений.

Після визначення комплексних критеріїв ефективності для вибору єдиного рішення задачі оптимізації варіантів розвитку необхідно шукати альтернативу, найбільш близьку до "ідеального" стану за всіма критеріями.

Розв'язання проблем багатокритеріальної алгоритмами та методами другої групи.

Найважливішим інструментом розв'язання багатокритеріальних задач алгоритмами та методами другої групи є принцип Еджворта-Парето (принцип Парето).

Застосування принципу Еджворта-Парето дозволяє з множини всіх можливих рішень виключити завідомо неприйнятні рішення, тобто ті, які ніколи не можуть виявитися обраними, якщо вибір здійснюється досить «розумно». Після такого виключення залишається множина рішень, яке називають множиною Парето або областю компромісних рішень.

Це рішення, як правило, є досить широким і в процесі прийняття рішень неминуче постає питання про те, яке саме можливе рішення вибрати серед парето-оптимальних?

До теперішнього часу таких методів, схем і підходів пошуку компромісу налічується не один десяток. Слід, однак, відзначити, що вони, як правило, мають слабке теоретичне обгрунтування і носять, в основному, евристичний характер.

А найголовніше - автори запропонованих методів не можуть чітко описати клас тих завдань вибору, для вирішення яких застосування даного методу гарантовано призводить до дійсно найкращому рішенню.

Парето-оптимальне рішення - це таке допустиме рішення, яке не може бути покращено (збільшено) ні по одному з наявних критеріїв без погіршення (зменшення) по якомусь хоча б одному іншим критерієм.

Інакше кажучи, вважаючи за краще одному парето-оптимального рішення інше парето-оптимальне рішення, ми будемо змушені йти на певний компроміс, погоджуючись на деяку втрату хоча б за одним критерієм (отримуючи, зрозуміло, певний виграш, в крайному разі, по якомусь іншому критерію). З цієї причини множина Парето нерідко називають множиною компромісів.

Поняття оптимальності по Парето відіграє важливу роль в математичній економіці. Саме в цій області часто замість парето-оптимальності використовують найменування ефективне рішення і множина ефективних рішень. Тим самим, парето-оптимальність і ефективність в математичній економіці нерідко виявляються синонімами.

Представимо поняття множини Парето на задачі двокритеріальної:

$$К_1 = F(x_i) \rightarrow \max;\ К_2 = F(x_2) \rightarrow \max;$$

$$\{G_i\ (xi) \geq 0$$

На рис. 1. 3 представлено область Парето рішень заданої задачі. Стрілочкою показано криву Парето точок, з яких і вибирається компромісне рішення.

Зворотне перетворення на область допустимих рішень по вибранному компромісному рішенню й буде представляти вектор рішень вхідної задачі.

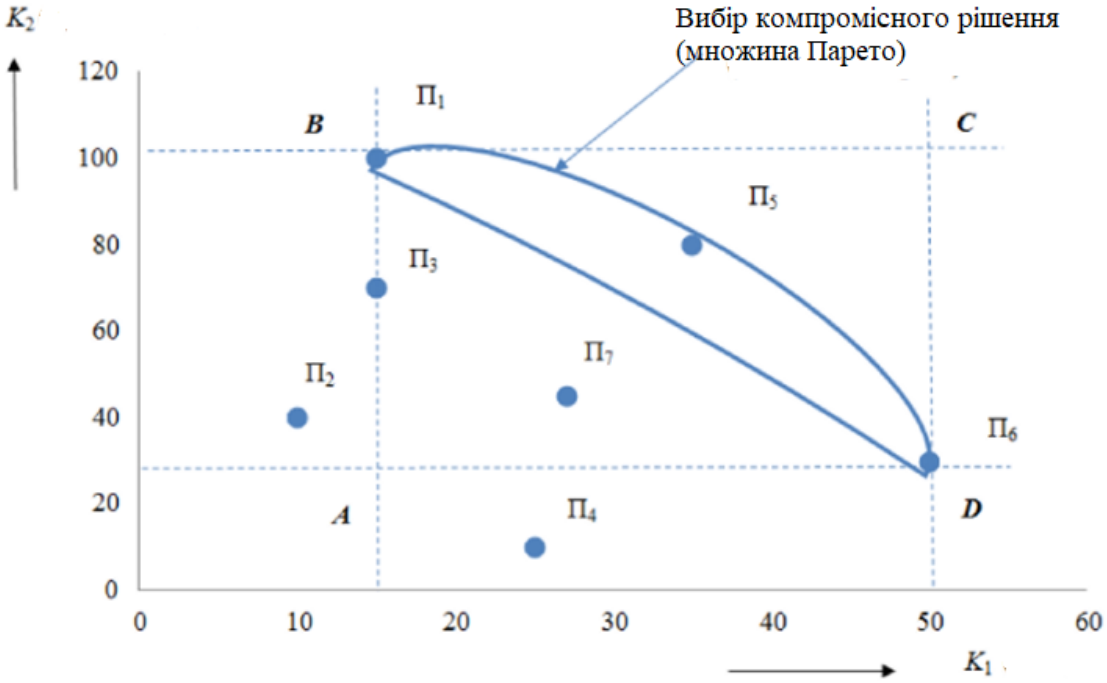

Рисунок 1.3 – Вибір компромісного рішення на множині Парето

Аналіз існуючих рішень приводить до наступного висновку.

На сьогоднішній день існують наступні проблеми багатокритеріальної оптимізації.

Перша проблема пов'язана з вибором принципу оптимальності, який строго визначає властивості оптимального рішення і відповідає на питання, в якому сенсі оптимальне рішення вигідно відрізняється від інших допустимі рішення. На відміну від завдань однокритеріальної оптимізації, які мають тільки один принцип оптимальності f (Xo) ≥f (x), в даному випадку є велика кількість різних принципів, і кожен принцип може призводити до вибору різних оптимальних рішень. Це пояснюється тим, що доводиться порівнювати вектори ефективності на основі деякої схеми компромісу. З проведеного аналізу найбільш вживаним в цьому плані є принцип оснований на компромісних рішеннях по Парето.

Друга проблема пов’язана з так званою нормалізацією векторного критерію ефективності. Вона визнана тим, що локальні критерії оптимальності мають різні масштаби вимірювання, а потрібно для порівнянь їх нормалізувати.

Третя проблема визначається з розстановкою пріоритетів по критеріям. Ранжування відбувається для встановлення важливості виконання цілей.

І основна проблема пов’язана з вибором алгоритму чи методу до розв’язання конкретної задачі багатокритеріальної оптимізації, бо не існує єдиного загального алгоритму, що дозволяє вибрати яка саме модель багатокритеріальності приведе до найкращого результату в конкретній ситуації.

1.2 Постановка задачі

Нехай задано багатокритеріальну задачу оптимізації.

Поставимо наступне завдання дослідження.

А*) Створити інформаційне та програмне забезпечення рішення задач трьохкритеріальної оптимізації:*

Б) Адаптувати метод Монте-Карло для рішення поставленого завдання.

В) Провести тестові випробування комп’ютерної реалізації проекту на завданнях багатокритеріальної оптимізації.

2 ВИБІР АЛГОРИТМІВ РІШЕННЯ ПРОБЛЕМИ

2.1 Короткий огляд відомих рішень

В однокритеріальних задачах програмування вибір рішення відбувається з одною метою (критерієм) і тим самим єдиною цільовою функцією. Навіть, якщо задача нелінійної оптимізації – ціль єдина.

Багатокритеріальність вносить в процес пошуку оптимального рішення багато проблем, основна з них та, що якість рішення оцінюється за багатьма критеріями [1-3].

Виникає питання, якщо вибір найкращого рішення є нетривіальним завданням, з'являється нова проблема: що слід розуміти під оптимальним рішенням? Справа в тому, що об'єктивно невідомо, яке рішення краще, якщо критеріїв багато і вони, можливо, «конфліктуючі». Зокрема, незрозуміло, яке рішення краще, якщо фірма одночасно прагне максимізувати прибуток і мінімізувати витрати. Тому потрібно шукати компромісне рішення, що враховує важливість кожної цільової функції.

Проблему вибору рішення з парето-оптимальних можна решити, наприклад, використовуючи метод цільового програмування. При такому підході вдається побудувати одне рішення, яке є парето-оптимальним. Слід зазначити, що цільове програмування - це не єдиний метод знаходження одного парето-оптимального рішення. Зокрема, є спеціальна теорія арбітражних схем, вирішальна дану проблему.

У разі лінійної багатокритеріальної задачі має сенс говорити про екстремальні ефективних (парето-оптимальних) рішеннях. Таких рішень кінцеве число, що істотно спрощує вирішення проблеми вибору.

Нехай X позначає множину допустимих рішень деякої задачі, x є X - допустиме рішення. Припустимо, що кожне рішення x є X оцінюється по n критеріям ($n \geq 2$).

Є надзвичайно корисна конструкція для вирішення багатокритеріальних завдань - поняття оптимального по Парето або ефективного вирішення.

Нехай H_i (x), x є X – функція дійсних значень, значеннями якої є оцінки рішення x є X за i-м критерієм.

Тоді вектор H (x) = (H_1 (x), …, H_i (x), …, Hn (x)), x є X - набір оцінок рішення x є X за всіма критеріями. Припустимо, що ступінь перевагу рішення x є X зростає зі зростанням компонент вектора H, тобто, чим більше значення Hi (x), тим краще рішення x за критерієм i, i є 1, n.

Рішення x* є X називається парето-оптимальним (оптимальвим по Парето, ефективним), якщо не існує іншого рішення xє X, для якого

$$H_i(x) \geq H_i(x^*), i = \overline{1,n},$$
$$\exists i_0 : H_{i_0}(x) > H_{i_0}(x^*).$$

(2.1)

Іншими словами, якщо x* є X - парето-оптимальне рішення, то не існує іншого рішення x є X, яке перевершує x* хоча б за одним критерієм, а за іншими критеріями не гірше.

 Перевагою концепції ефективного рішення є те, що ефективні рішення існують в практично значущих класах задач.

 Недолік поняття парето-оптимального рішення полягає в тому, що воно, як правило, не дозволяє знайти єдиного вирішення проблеми, можна отримати лише множину ефективних рішень.

Наприклад, задано трьохкритеріальну задачу (див. табл 2.1)

Таблиці 2.1Вибір автомобіля

	VW Golf	Opel Astra	Ford Focus	Toyota Corolla
Цена (1000 Euro)	16.2	14.9	14.0	15.2
Расход топлива (на 100 км)	7.2	7.0	7.5	8.2
Мощность (kW)	66.0	62.0	55	71

Виникає питання - Який автомобіль вибрати, щоб він був потужним, недорогим, з малою витратою палива?

Серед відомих методів вирішення представимо метод цільового програмування.

В основі методу цільового програмування для розв'язання багатокритеріальних задач лежить впорядкування критеріїв (цілей) за ступенем важливості.

Вихідна задача вирішується шляхом послідовного вирішення низки завдань з однією цільовою функцією таким чином, що рішення задачі з менш важливою на меті не може погіршити оптимального значення цільової функції з більш високим пріоритетом.

Основна відмінність цільового програмування:

 багато цілі формалізуються не як цільові функції, аяк обмеження в інший більш загальної моделі.

З цією метою вводяться:

 передбачувані кількісні значення цільових функцій;

 змінні відхилення які характеризують ступінь досягнення поставлених цілей для даного рішення.

Основними алгоритмами в методі цільового програмування є

- метод вагових коефіцієнтів;
- метод пріоритетів.

Оптимальне рішення, обирається на основі багатокритеріального підходу незалежно від обраного принципу оптимальності, завжди має належати області компромісів.

Інакше воно може бути покращено і, отже, не є оптимальним. Таким чином, область компромісів є область потенційно оптимальних компромісів. Звідси випливає, що при виборі рішення по векторному критерію можна обмежити пошук оптимального рішення областю компромісів, яка, як правило, значно вужче всій області можливих рішень X.

Методи порівняння векторних оцінок з використанням додаткової інформації:

- однокрокові методи:
- - евристичні (не мають строгого обґрунтування, застосовуються тільки

- для конкретних типів задач);
    - метод головного критерію;
- - аксіоматичні (базуються на деякій системі аксіом).
- багатокрокові методи – достатньо складні для розуміння;
- багатокритеріальну задачу прийняття рішень на множині Парето
- можна звести до однокритеріальної, вводячи деякий узагальнений
- критерій Z* як функцію від попередніх окремих критеріїв.

Узагальнений критерій Z* в літературі ще називають функцією корисності.

Процес зведення багатокритеріальної задачі до однокритеріальної називається згорткою.

Рисунок 2.1 – Методи згорток за критеріями

В основу методу, що буде застосовуватись в наших дослідженнях лежить ідея наближення по всім критеріям з застосуванням чисельного рішення.

Нехай дана задача багатокритеріальної оптимізації

$$\begin{cases} f_1(x) \to \max \\ f_2(x) \to \max \\ .. \\ f_k(x) \to \max \\ x \in X \end{cases} \quad \textbf{(2.2)}$$

З заданими обмеженнями

$$\begin{cases} G(\ (x1, x2, x3, \dots, xn) \geq 0, \\ Q(\ (x1, x2, x3, \dots, xn) \leq 0, \\ \quad (x1, x2, x3, \dots, xn) \geq 0 \end{cases} \quad (2.3)$$

Будемо шукати оптимальні рішення по кожному з критеріїв, застосовуючи чисельний метод рішення, а потім знайдемо з цих рішень наближення, що буде компромісним по всім критеріям одночасно.

2.2 Адаптація методу Монте-Карло для рішення задач багатокритеріальної оптимізації

Метод Монте-Карло можна розглядати як д статистичних випробувань для рішення різного класу, в яких використовуються фундаментальні випадкові вибірки. В рамках багатокритеріальної оптимізації, ми його застосуємо для вирішення оптимізації за кожним заданим критерієм.

В основі застосування для рішення будь-якого завдання алгоритмом Монте-Карло лежить генератор випадкових чисел: процедура, яка створює нескінченний потік випадкових величин, які є незалежними та однаково розподіленими відповідно до деякого розподілу ймовірностей.

Генератори засновані на простих алгоритмах, які можна легко реалізувати на комп'ютері. Такі алгоритми зазвичай можна представити у вигляді кортежу (S, f, μ, U, g), де

- S - скінченна сукупність станів,
- f - функція від S до S,
- μ - розподіл ймовірностей на S,
- U - вихідний простір; для рівномірного генератора випадкових чисел U дорівнює інтервал (0, 1), і ми будемо вважати це відтепер, якщо не інше вказано,
- g - це функція від S до U.

Застосуємо алгоритм Монте-Карло для рішення завдання багатокритеріальної оптимізації в плані рішення задач оптимальності по кожному критерію. Блок схема адаптації МСМ наведена на рисунку 2.3.

Опишемо основні його складові.

1. Задаємо вхідну інформацію : функцію цілі та обмеження.
2. Генеруємо за допомогою генератора випадкових величин точки.
3. Виконаємо процедуру, визначаючу чи входять генеровані точки до області допустимих рішень. З цією метою ми проходимо «сито» обмежень, що задає область допустимих рішень.

4. Для кожної з точок, що ввійшли в область допустимих рішень (пройшли «сито» обмежень) знаходимо значення функції цілі.
5. Запускаємо процедуру пошуку точки, що забезпечує шуканий екстремум.
6. Запускаємо процедуру звуження області генерування випадкових точок, якщо це потрібно.
7. Проводимо кількість статистичних випробувань, збільшуючи їх до досягнення заданої точності розрахунків ε.
8. Точність розрахунку перевіряємо за формулою
   $F_{n+1} - F_n < \varepsilon$. Ця нерівність одночасно служить виходом із циклу .
9. При досягненні заданої точності обчислень закінчуємо роботу алгоритму.

3 РОЗРОБКА ІНФОРМАЦІЙНОГО ТА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ТА ТЕСТОВІ РОЗРАХУНКИ

3.1 Комп’ютерна реалізація алгоритмів та основні її складові

Комп’ютерна реалізація обраних алгоритмів проводилися на комп'ютері з наступною конфігурацією:

- ✓ ПРОЦЕССОР Intel (R) Celeron CPU 3050 @ 1.6 GHz;
- ✓ ОПЕРАТИВНА ПАМ'ЯТЬ (ОРЕ) 2.0 ГБ;
- ✓ ОПЕРАЦІЙНА СИСТЕМА Windows 10;
- ✓ Тип системи 32-розрядна операційна система.

Вхідна задача багатокритеріальної оптимізації задавалась в загальному разі:

$$\begin{cases} f_1(x) \to \max \\ f_2(x) \to \max \\ \dots \\ f_k(x) \to \max \\ x \in X \end{cases} \qquad (3.1)$$

З заданими обмеженнями

$$\begin{cases} G(\,(x1, x2, x3, \dots, xn) \geq 0, \\ Q(\,(x1, x2, x3, \dots, xn) \leq 0, \\ \quad (x1, x2, x3, \dots, xn) \geq 0 \end{cases} \qquad (3.2)$$

В якості мови програмування для комп’ютерної реалізації обрано мову високого рівня Java.

Застосування методу Монте-Карло визначимо як окремий клас

```
public class MonteCarlo {
  public static void main(String[] args) throws IOException {
    /* Результат виводимо до файлу Monte_Carlo_Results.txt */
```

Генерування точок області генератором випадкових чисел відбувалось за правилом

```
    /* Згенеруємо 100 точок */
```

```
        double[] result = monteCarloMethod(100);
        writer.append(100 + " ")
                .append(Double.toString(result[0])).append(" ").append(Dou-
ble.toString(result[1])).append(" ")
                .append(Double.toString(firstCriteria(result[0], result[1]))).append(" ")
                .append(Double.toString(secondCriteria(result[0], result[1]))).append(" ")
                .append(Double.toString(thirdCriteria(result[0], result[1]))).append(" ")
                .append("\n");
```

Рішення по кожному критерію методом Монте Карло відбувалось в наступному блоці програми

```
    /* Метод Монте Карло */
    static double[] monteCarloMethod(int amountOfPoints) {
        boolean isCriteriaWasCalculated = false;
        /* Масив оптимальних значень критеріїв */
        double[] criteriaValues = new double[3];
        /* Масив координат точок, які відповідають оптимальним значенням */
        double[][] x1x2Criteria = new double[3][2];
        for (int i = 0; i < amountOfPoints; i++) {
            /* Генеруємо точку */
            double x1 = getRandomNumber(0, 13);
            double x2 = getRandomNumber(0, 10.5);
            /* Перевіряємо чи потрапляє точка у область */
            if (isCoordinatesInArea(x1, x2)) {
                /* Обчислюємо значення критеріїв */
                double firstCriteriaValueRes = firstCriteria(x1, x2);
                double secondCriteriaValueRes = secondCriteria(x1, x2);
```

```
            double thirdCriteriaValueRes = thirdCriteria(x1, x2);
                if (!isCriteriaWasCalculated)
```

Вибір компромісного рішення відбувався за правилом

```
/* Знаходимо середнє значення координат точок, що відповідають оптимальним значенням кожного критерію */
    double x1 = (x1x2Criteria[0][0] + x1x2Criteria[1][0] + x1x2Criteria[2][0]) / 3;
    double x2 = (x1x2Criteria[0][1] + x1x2Criteria[1][1] + x1x2Criteria[2][1]) / 3;
    return new double[]{x1, x2};
  }
```

Перевірка потрапляння точки в задану область відбувалася в наступному фрагменті

```
  /* Перевіряє умови потрапляння точки в область */
  static boolean isCoordinatesInArea(double x1, double x2) {
    return x1 >= 0 && x2 >= 0 && (2 * x1 + 3 * x2 >= 6) && (3 * x1 - 2 * x2 <= 18) && (-1 * x1 + 2 * x2 <= 8);
  }
```

3.2 Тестові розрахунки та порівняльний аналіз

Для перевірки працездатності комп'ютерної реалізації виконаємо тестові розрахунки на наступному завданні.

Розв'язати трьохкритеріальну задачу за критеріями

$F_1 = f_1(x_1, x_2) = x_1^2 + x_2^2 + 25 - 8x_1 - 6x_2 \rightarrow \max$

$F_2 = x_2 - x_1 \rightarrow \max$

$F_3 = (x_1 - 1)^2 + (x_2 - 1)^2 \rightarrow \min$

З заданими обмеженнями

$$\begin{cases} 2x_1 + 3x_2 \geq 6 \\ 3x_1 - 2x_2 \leq 18 \\ -x_1 + 2x_2 \leq 8 \\ x_1, x_2 \geq 0 \end{cases}$$

Рішення проведемо за алгоритмом, що представлений в п. 2.2

1. Розіб'ємо завдання на три однокритеріальні задачі та розв'яжемо кожну з них за алгоритмом Монте-Карло.
2. Множина розвозів надана в табл. 3.1 .

Таблиця 3.1 Знаходження компромісного рішення та значення функцій цілі в точках компромісу

number of points	x1	x2	first criteria	second criteria	third criteria
1(100)	3,997045133	4,867945138	3,489227768	0,870900005	23,94327911
2(500)	4,748293975	5,159287624	5,222466915	0,410993649	31,34938126
3(1000)	4,962946956	5,265757048	6,060921838	0,302810092	33,90163176
4(10000)	4,761551005	5,232824667	5,565465927	0,471273661	32,06607063
5(100000)	4,709543871	5,226829388	5,46222163	0,517285517	31,62680241

При застосуванні методу Монте-Карло ми варіювали кількість статистичних іспитів. В даному разі це кількість точок генерації6 100, 500, 1000, 10000 та 100000. Ці генерації пронумеруємо в порядковому збільшенні номерами 1, 2… і т.д. Така зручність нам знадобиться при побудові графіків функцій.

3. Позначимо вектори рішень за критеріями наступне (x_1^1, x_2^1); (x_1^2, x_2^2), (x_1^3, x_2^3).
4. Знаходимо компромісне рішення (x_1^*, x_2^*) за формулами

$X_1^* = (x_1^1 + x_1^2 + x_1^3)/3.$

$X_2^* = (x_2^1 + x_2^2 + x_2^3)/3.$

Точка К, що має координати (x_1^*, x_2^*) і є шуканим результатом рішення задачі багатокритеріальної оптимізації.

5. Перерахуємо значення усіх трьох критеріїв в компромісній точці рішення та будуємо графіки значень цих критеріїїв в залежності від кількості проведених статистичних випробувань метода Моете-Карло.

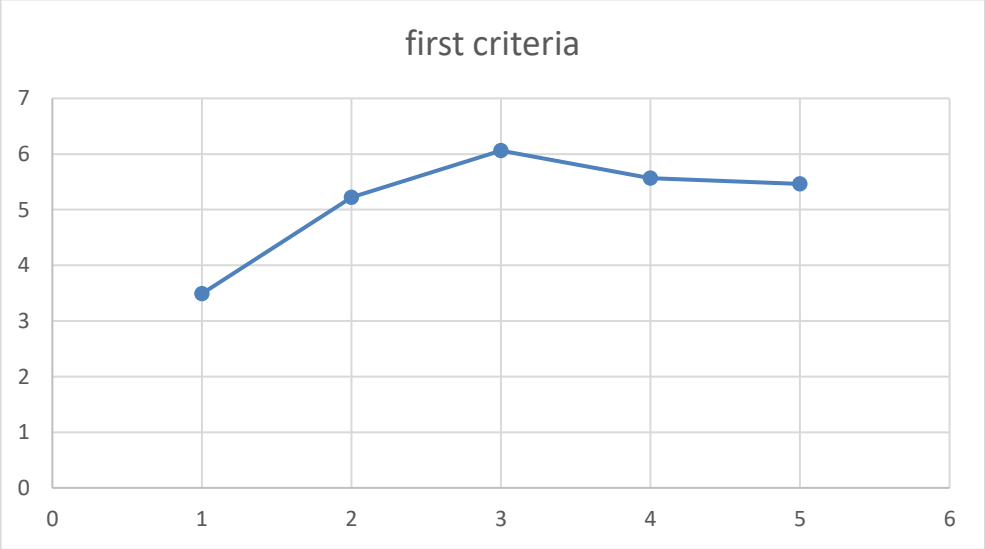

Рисунок 3.1 – Залежність критерію F_1 від кількості точок статистичних випробувань

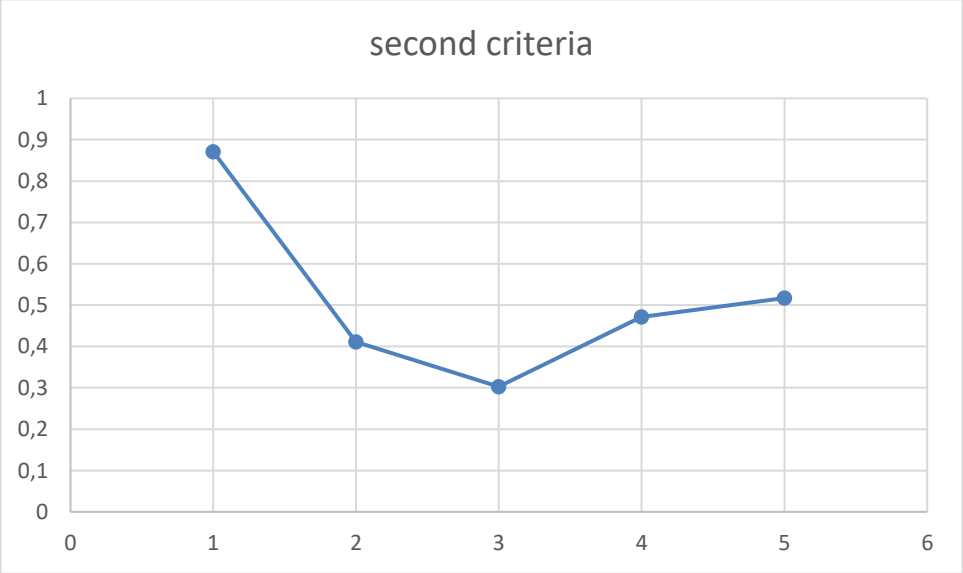

Рисунок 3.2 – Залежність критерію F_2 від кількості точок статистичних випробувань

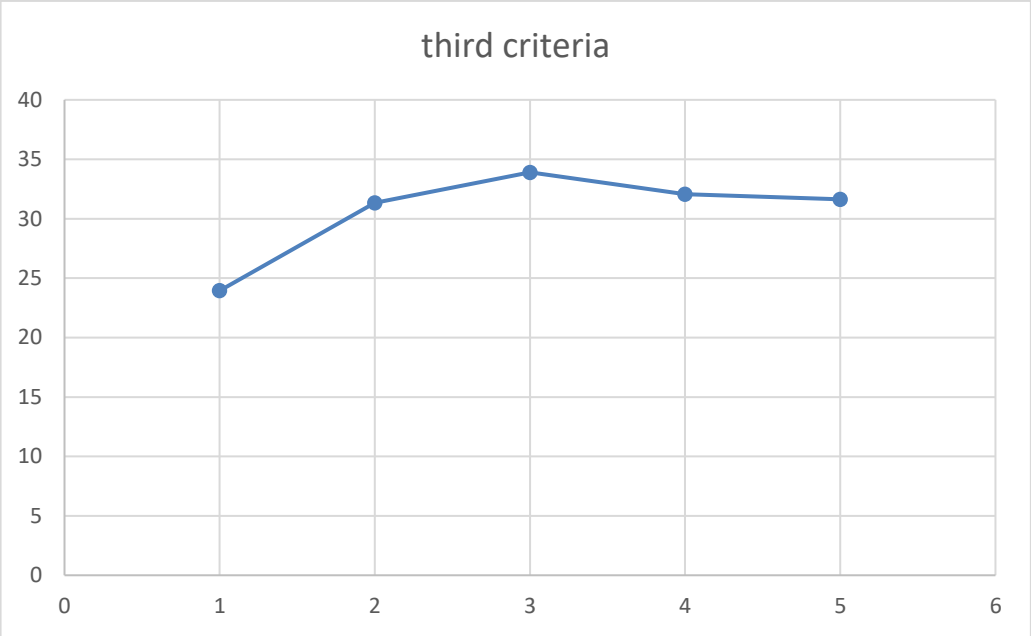

Рисунок 3.3 – Залежність критерію F_3 від кількості точок статистичних випробувань

ВИСНОВКИ

В магістерській роботі проведені дослідження по розв’язанню задач багатокритеріальної оптимізації. При рішенні застосований алгоритм Монте-Карло, що послуговує одержанню чисельного розв’язку з деякою точністю. Створено компютерну реалізацію на мові програмування Java.

За результатами досліджень можна зробити наступні висновки.

1. Адаптація алгоритму Монте-Карло для розв’язання задач багатокритеріальної оптимізації дозволяє одержати рішення, що збігаються з розв’язками за іншими алгоритмами.

2. В результаті виконання алгоритму ми одержуємо компромісне рішення, що задовольняє усім заданим критеріям оптимізації.

3. Тестові розрахунки показали, шо даний підхід може застосовуватись при рішенні задач багатокритеріальної оптимізації.

СПИСОК ЛІТЕРАТУРИ

1. Файнзільберг Л. С., Жуковська О. А., Якимчук В. С. Теорія прийняття рішень. – Київ: Освіта України, 2018. – 246 с.
2. David G. Luenberger, Yinyu Ye Linear and Nonlinear Programming. - Springer, 2015. - 546 p.
3. Математичні методи дослідження операцій : підручник / Є. А. Лавров, Л. П. Перхун, В. В. Шендрик та ін. – Суми : Сумський державний університет, 2017. – 212 с.
4. Таха Хэмди А. Исследование операций/ Таха Хэмди А. – Москва: Вильямс, 2016. – 912 с.
5. Льовкін В. Програмна реалізація багатокритеріальних методів прийняття рішень// “COMPUTER SCIENCE & ENGINEERING 2009” (CSE-2009), 14-16 MAY 2009, LVIV, UKRAINE с. 243-246
6. Колесніков К. В., Карапетян А. Р., Царенко Т. А. Генетичні алгоритми для задач багатокритеріальної оптимізації в мережах адаптивної маршрутизації даних// Вісник НТУ “ХПІ». 2013. №56(1029) с. 44-50
7. Міщенко П. М., Шаповалов С. П. Рішення багатокритеріальних задач генетичним алгоритмом// Матеріали XXIII міжнародної конференції з автоматичного управління (Автоматика 2016), Суми, СумДУ, 2016. – с. 35-36.
8. Шаповалов С. П. Застосування методу Монте-Карло в обчислювальних моделях// Матеріали та програма міжнародної науково-технічної конференції студентів та молодих вчених (ІМА-2020), Суми, СумДУ, 2020. – с. 84.

ДОДАТОК

```
import java.io.BufferedWriter;

import java.io.FileWriter;

import java.io.IOException;


public class MonteCarlo {

    public static void main(String[] args) throws IOException {

        /* Результат виводимо до файлу Monte_Carlo_Results.txt */

        BufferedWriter writer = new BufferedWriter(new FileWriter("Monte_Carlo_Results.txt", true));

        writer.append("number of points | x1 | x2 | first criteria | second criteria | third criteria\n");

        /* Згенеруємо 100 точок */

        double[] result = monteCarloMethod(100);

        writer.append(100 + " ")

                .append(Double.toString(result[0])).append(" ").append(Double.toString(result[1])).append(" ")

                .append(Double.toString(firstCriteria(result[0], result[1]))).append(" ")

                .append(Double.toString(secondCriteria(result[0], result[1]))).append(" ")

                .append(Double.toString(thirdCriteria(result[0], result[1]))).append(" ")

                .append("\n");

        /* Згенеруємо 500 точок */

        result = monteCarloMethod(500);

        writer.append(500 + " ")

                .append(Double.toString(result[0])).append(" ").append(Double.toString(result[1])).append(" ")

                .append(Double.toString(firstCriteria(result[0], result[1]))).append(" ")
```

```
            .append(Double.toString(secondCriteria(result[0], result[1]))).append(" ")
            .append(Double.toString(thirdCriteria(result[0], result[1]))).append(" ")
            .append("\n");
        /* Згенеруємо 1000 точок */
        result = monteCarloMethod(1000);
        writer.append(1000 + " ")
            .append(Double.toString(result[0])).append(" ").append(Dou-
ble.toString(result[1])).append(" ")
            .append(Double.toString(firstCriteria(result[0], result[1]))).append(" ")
            .append(Double.toString(secondCriteria(result[0], result[1]))).append(" ")
            .append(Double.toString(thirdCriteria(result[0], result[1]))).append(" ")
            .append("\n");
        /* Згенеруємо 10000 точок */
        result = monteCarloMethod(10000);
        writer.append(10000 + " ")
            .append(Double.toString(result[0])).append(" ").append(Dou-
ble.toString(result[1])).append(" ")
            .append(Double.toString(firstCriteria(result[0], result[1]))).append(" ")
            .append(Double.toString(secondCriteria(result[0], result[1]))).append(" ")
            .append(Double.toString(thirdCriteria(result[0], result[1]))).append(" ")
            .append("\n");
        /* Згенеруємо 100000 точок */
        result = monteCarloMethod(100000);
        writer.append(100000 + " ")
            .append(Double.toString(result[0])).append(" ").append(Dou-
ble.toString(result[1])).append(" ")
            .append(Double.toString(firstCriteria(result[0], result[1]))).append(" ")
```

```
            .append(Double.toString(secondCriteria(result[0], result[1]))).append(" ")
            .append(Double.toString(thirdCriteria(result[0], result[1]))).append(" ")
            .append("\n");
    writer.close();
  }

  /* Метод Монте Карло */
  static double[] monteCarloMethod(int amountOfPoints) {
    boolean isCriteriaWasCalculated = false;
    /* Масив оптимальних значень критеріїв */
    double[] criteriaValues = new double[3];
    /* Масив координат точок, які відповідають оптимальним значенням */
    double[][] x1x2Criteria = new double[3][2];
    for (int i = 0; i < amountOfPoints; i++) {
      /* Генеруємо точку */
      double x1 = getRandomNumber(0, 13);
      double x2 = getRandomNumber(0, 10.5);
      /* Перевіряємо чи потрапляє точка у область */
      if (isCoordinatesInArea(x1, x2)) {
        /* Обчислюємо значення критеріїв */
        double firstCriteriaValueRes = firstCriteria(x1, x2);
        double secondCriteriaValueRes = secondCriteria(x1, x2);
        double thirdCriteriaValueRes = thirdCriteria(x1, x2);
        if (!isCriteriaWasCalculated) {
          /* Якщо це перше обчислення значень критеріїв, то записуємо їх у
відповідні змінні */
```

```
            criteriaValues[0] = firstCriteriaValueRes;
            criteriaValues[1] = secondCriteriaValueRes;
            criteriaValues[2] = thirdCriteriaValueRes;
            x1x2Criteria[0][0] = x1;
            x1x2Criteria[0][1] = x2;
            x1x2Criteria[1][0] = x1;
            x1x2Criteria[1][1] = x2;
            x1x2Criteria[2][0] = x1;
            x1x2Criteria[2][1] = x2;
            isCriteriaWasCalculated = true;
        } else {
            /* Якщо критерії вже обчислювалися, то порівнюємо нові значення з оптимальними,
            що були отримані раніше і оновлюємо оптимальні значення у разі необхідності */
            if (firstCriteriaValueRes > criteriaValues[0]) {
                criteriaValues[0] = firstCriteriaValueRes;
                x1x2Criteria[0][0] = x1;
                x1x2Criteria[0][1] = x2;
            }
            if (secondCriteriaValueRes > criteriaValues[1]) {
                criteriaValues[1] = secondCriteriaValueRes;
                x1x2Criteria[1][0] = x1;
                x1x2Criteria[1][1] = x2;
            }
            if (thirdCriteriaValueRes < criteriaValues[2]) {
```

```
                criteriaValues[2] = thirdCriteriaValueRes;
                x1x2Criteria[2][0] = x1;
                x1x2Criteria[2][1] = x2;
            }
        }
    }
}
/* Знаходимо середнє значення координат точок, що відповідають оптимальним значенням кожного критерію */
double x1 = (x1x2Criteria[0][0] + x1x2Criteria[1][0] + x1x2Criteria[2][0]) / 3;
double x2 = (x1x2Criteria[0][1] + x1x2Criteria[1][1] + x1x2Criteria[2][1]) / 3;
return new double[]{x1, x2};
}

/* Перевіряє умови потрапляння точки в область */
static boolean isCoordinatesInArea(double x1, double x2) {
return x1 >= 0 && x2 >= 0 && (2 * x1 + 3 * x2 >= 6) && (3 * x1 - 2 * x2 <= 18) && (-1 * x1 + 2 * x2 <= 8);
}

/* Перший критерій */
static double firstCriteria(double x1, double x2) {
return Math.pow((x1 - 4), 2) + Math.pow((x2 - 3), 2);
}

/* Другий критерій */
```

```
    static double secondCriteria(double x1, double x2) {
        return x2 - x1;
    }

    /* Третій критерій */
    static double thirdCriteria(double x1, double x2) {
        return Math.pow((x1 - 1), 2) + Math.pow((x2 - 1), 2);
    }

    /* Генерує випадкове число */
    static double getRandomNumber(double min, double max) {
        return ((Math.random() * (max - min)) + min);
    }
}
```