МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК

ВИПУСКНА РОБОТА

на тему:

«Адаптація методу Монте-Карло для рішення задач лінійного та нелінійного програмування»

Завідувач випускаючої кафедри — **Довбиш А.С.**

Керівник роботи — **Шаповалов С.П.**

Студентки групи ІН – 63 — **Сірик Є.В.**

СУМИ 2020

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ

Кафедра комп'ютерних наук

Затверджую_____

Зав. кафедрою Довбиш А.С.

"_____" _____2020 р.

ЗАВДАННЯ

до випускної роботи

Студентки четвертого курсу, групи ІН-63 спеціальності "Інформатика" денної форми навчання Сірик Євгенії Володимирівни.

Тема: "Адаптація методу Монте-Карло для рішення задач лінійного та нелінійного програмування"

Затверджена наказом по СумДУ

№_____ от _____2020 р.

Зміст пояснювальної записки: 1) аналіз проблеми та постановка задачі; 2) вибір метода розв'язання задачі; 3) розробка інформаційного і програмного забезпечення системи

Дата видачі завдання "_____"_____2020 р.

Керівник випускної роботи _____ Шаповалов С.П.

Завдання прийняла до виконання _____Сірик Є.В.

РЕФЕРАТ

Записка: 53 стор., 10 рис., 1 табл., 1 додаток, 7 джерел.

Об’єкт дослідження — задачі лінійного та нелінійного програмування.

Мета роботи — адаптація методу Монте-Карло для вирішення задач лінійного та нелінійного програмування.

Методи дослідження — Математичне моделювання, метод Монте-Карло, комп’ютерна реалізація алгоритмів.

Результати — розроблено інформаційне та програмне забезпечення вирішення задач лінійного та нелінійного програмування методом Монте-Карло. Проведені тестові розрахунки застосовності методу Монте-Карло для задач лінійного та нелінійного програмування. Для комп’ютерної реалізації застосовано алгоритмічну мову C++.

МЕТОД МОНТЕ-КАРЛО, ЛІНІЙНЕ ПРОГРАМУВАННЯ,
СИПЛЕКС-МЕТОД, НЕЛІНІЙНЕ ПРОГРАМУВАННЯ,
ПРОГРАМНА РЕАЛІЗАЦІЯ

ЗМІСТ

ВСТУП

Основні проблеми в життєдіяльності та виробничих сферах, що виникали й виникають перед людством, в рамках обмежених ресурсів – це проблеми, які можна віднести до рішень задач оптимізації та лінійного й нелінійного програмування [1, 2]. Зрозуміло, що до «комп'ютерної ери» основними методами досліджень в цих сферах були аналітичні розрахунки, або такі численні методи, які б не потребували при своїй реалізації в громістких обчислень, бо на чому їх проводити?

З розвитком комп'ютерної техніки зростає в вирішенні цих проблем інтерес до вибору інструментарію, що опирається на методи чисельного моделювання. Серед останніх виділяються два напрями : 1) Методи, основані на еволюційній парадигмі (генетичні алгоритми); 2) Методи, основані на статистичній парадигмі (алгоритми Монте-Карло) [3-7].

Методи Монте-Карло, або експерименти Монте-Карло, являють собою широкий клас обчислювальних алгоритмів, які покладаються на багаторазову випадкову вибірку для отримання чисельних результатів. Основна концепція полягає в використанні випадковості для вирішення проблем, які можуть бути в принципі детермінованими. Методи Монте-Карло перевертають звичайну проблему статистики: замість того, щоб оцінювати випадкові величини детермінованим способом, випадкові величини використовуються для оцінки детермінованих величин. В принципі, методи Монте-Карло можуть бути використані для вирішення будь-якої проблеми, що має ймовірнісну інтерпретацію.

Методи Монте-Карло вже відомо використовуються в фізичних і математичних задачах і найбільш корисні, коли важко або неможливо використовувати інші підходи[3-4].

Але в останні роки відбувся значний сплеск використання методів Монте-Карло в сугубо «нематематичних» дисциплінах, як то хімія, біологія [7], значний інтерес виник у використаннях їх в теорії ігор [4].

Основна суть у вирішенні будь-яких складних проблем методом Монте-Карло залишається незмінною – ми запускаємо (генеруємо) процес, дійство, вибір необхідних вар’їруємих параметрів, а надалі за вихідними результатами обираємо той, що потребує проблема.

1 ОГЛЯД ІСНУЮЧИХ РІШЕНЬ

1.1 Моделі лінійного та нелінійного програмування

Лінійне програмування - це проста техніка, де ми зображуємо складні стосунки за допомогою лінійних функцій, а потім знаходимо оптимальні точки. Важливе слово в попередньому реченні зображене. Реальні стосунки можуть бути набагато складнішими, але ми можемо спростити їх до лінійних стосунків.

Додатки лінійного програмування всюди навколо вас. Ви використовуєте лінійне програмування на особистому і професійному рівнях. Ви використовуєте лінійне програмування, коли їдете з будинку на роботу і хочете вибрати найкоротший шлях. Чи, коли у вас є проект, ви розробляєте стратегії, щоб ваша команда працювала ефективно і вчасно.

Приклад завдання лінійного програмування:

Припустимо, у постачальника FedEx є 6 посилок за день. Склад знаходиться в точці A. Шість пунктів призначення доставки задаються як U, V, W, X, Y і Z. Цифри в рядку x вказують відстань між містами. Щоб заощадити на паливі і часі, постачальник хоче вибрати найкоротший маршрут.

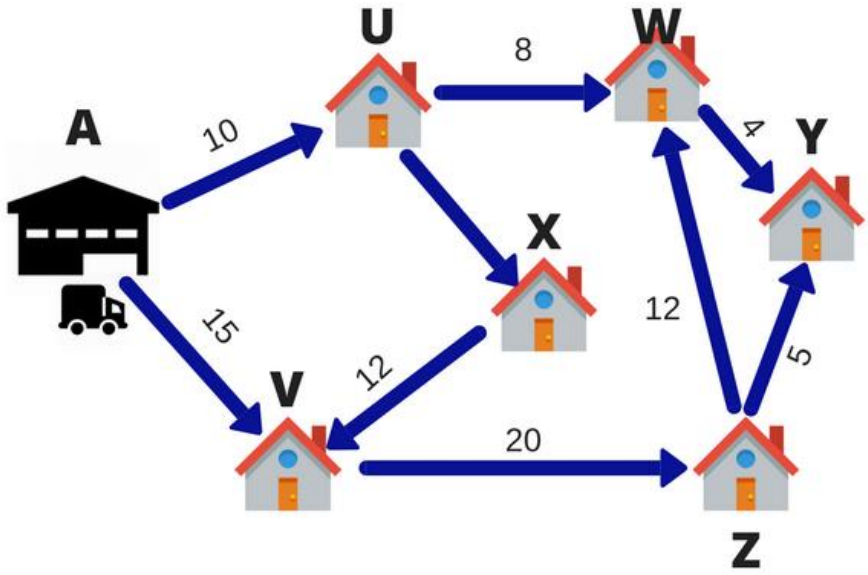

Рисунок 1.1 – Схема задачі ЛП

Таким чином, фахівець з доставки розрахує різні маршрути для відвідування усіх 6 пунктів призначення, а потім визначить найкоротший маршрут. Ця техніка вибору найкоротшого маршруту називається лінійним програмуванням.

В цьому випадку мета обличчя доставки полягає в тому, щоб доставити посилку вчасно за усіма 6 пунктами призначення. Процес вибору кращого маршруту називається Operation Research. Дослідження операцій - це підхід до ухвалення рішень, який включає набір методів для управління системою. У наведеному вище прикладі моєю системою була модель Delivery.

Лінійне програмування використовується для отримання найбільш оптимального рішення задачі із заданими обмеженнями. У лінійному програмуванні ми формулюємо нашу реальну проблему в математичну модель. Він включає цільову функцію, лінійні нерівності з урахуванням обмежень.

Чи являється лінійне представлення 6 точок вище представником реального світу? Так і ні. Це спрощення, оскільки реальний маршрут не буде прямою лінією. Швидше за все, він матиме декілька поворотів, розворотів, сигналів і пробок. Але з простим припущенням, ми значно зменшили складність проблеми і створюємо рішення, яке повинне працювати у більшості сценаріїв.

Формулювання проблеми - Давайте виготовимо декілька цукерок Приклад. Розглянемо компанію по виробництву шоколаду, яка робить тільки два типи шоколаду - A і B. Для обох цукерок потрібно тільки молоко і шоколад. Для виготовлення кожної одиниці A і B потрібно наступні кількості:

- Кожна одиниця A вимагає 1 одиницю молока і 3 одиниці Choco
- Кожна одиниця B вимагає 1 одиницю молока і 2 одиниці Choco

Кухня компанії складається з 5 одиниць молока і 12 одиниць шоколаду. З кожного продажу компанія отримує прибуток:

- 6 рупій за одиницю продані
- 5 рупій за одиницю B продані.

Тепер компанія хоче максимізувати свій прибуток. Скільки одиниць A і B він повинен зробити відповідно?

Рішення: Перше, що я збираюся зробити, це представити проблему у вигляді таблиці для кращого розуміння.

	Молоко	Choco	Прибыль на единицу
A	1	3	6 рупий
B	1	2	5 рупий
Всего	5	12	

Нехай загальна кількість одиниць, зроблених A, буде = X

Нехай загальна кількість одиниць, зроблених B, буде = Y

Тепер загальний прибуток представлений Z

Загальний прибуток, який компанія отримує, визначається як загальну кількість зроблених одиниць A і B, помножене на прибуток на одиницю продукції, рівну 6 і 5 рупія відповідно.

Прибуток: Макс Z = 6x + 5y

що означає, що ми повинні максимізувати Z.

Компанія постарається зробити якомога більше одиниць A і B, щоб максимізувати прибуток. Але ресурси Milk і Choco доступні в обмеженій кількості.

Згідно з приведеною вище таблицею, кожна одиниця A і B вимагає 1 одиницю молока. Загальна кількість доступного молока складає 5 одиниць. Щоб представити це математично

$$\mathbf{X + Y \leq 5}$$

Крім того, кожна одиниця A і B вимагає 3 одиниці & 2 одиниці Choco відповідно. Загальна кількість доступних Choco складає 12 одиниць. Щоб представити це математично

$$\mathbf{3x + 2y \leq 12}$$

Крім того, значення для одиниць A можуть бути тільки цілими числами.

Таким чином, у нас є ще два обмеження, $\mathbf{X \geq 0}$ **і** $\mathbf{Y \geq 0}$

Щоб компанія отримувала максимальний прибуток, повинні дотримуватися вище згадані нерівності.

Це називається формулюванням реальної проблеми в математичну модель.

Загальні терміни, використовувані в лінійному програмуванні:

- **Змінні рішення:** змінні рішення - це змінні, які визначатимуть мій висновок. Вони представляють моє остаточне рішення. Щоб розв'язати будь-яку проблему, нам спочатку треба визначити змінні рішення. Для наведеного вище прикладу загальна кількість одиниць для A і B, позначених X і Y відповідно, є моїми змінними рішення.
- **Мета Функція:** визначається як мета ухвалення рішень. У наведеному вище прикладі компанія хоче збільшити загальний прибуток, представлену Z. Таким чином, прибуток є моєю цільовою функцією.
- **Обмеження:** Обмеження - це обмеження або обмеження для змінних рішення. Вони зазвичай обмежують значення змінних рішення. У наведеному вище прикладі обмеження на доступність ресурсів Milk і Choco є моїми обмеженнями.
- **Обмеження позитивності:** для усіх лінійних програм змінні рішення завжди повинні набувати ненегативних значень. Це означає, що значення змінних рішення мають бути більші або рівні 0.

Процес формулювання завдання лінійного програмування

Давайте подивимося на етапи визначення завдання лінійного програмування загалом:

1. Визначте змінні рішення;
2. Напишіть цільову функцію;
3. Згадати обмеження;
4. Явно сформулюйте обмеження позитивності.

Щоб завдання було завданням лінійного програмування, змінні рішення, цільова функція і обмеження мають бути лінійними функціями.

Якщо усі три умови виконуються, це називається **завданням лінійного програмування .**

Хоча модель лінійного програмування відмінно працює у багатьох ситуаціях, деякі проблеми не можуть бути точно змодельовані без урахування нелінійних компонентів. Одним з прикладів буде ізотермічна задача: визначити форму кривої замкнутої площини, що має задану довжину і охоплює максимальну площу. Рішення, але не доказ, був відомий Паппус Александрійський.

Розділ математики, відомий як варіаційне числення розпочалося із спроб довести це рішення разом із завданням в 1696 році швейцарського математика Йоганн Бернулли, щоб знайти криву, яка мінімізує час, необхідний для ковзання об'єкту, тільки під дією сили тяжіння, між двома невертикальними точками. (Рішенням є брахістохрон .)

Окрім Йоганна Бернулли, його брата Якоба Бернулли, німця Готфрида Вільгельма Лейбніца і англійця Ісаака Ньютона усі вони надали правильні рішення. Зокрема, підхід Ньютона до рішення грає фундаментальну роль у багатьох нелінійних алгоритмах.

Інші чинники, що впливають на розвиток нелінійного програмування, такі як опуклий аналіз, теорія двоїстості і теорія управління, розроблений значною мірою після 1940 року.

Для завдань, які включають обмеження, а також цільову функцію, умови оптимальності, виявлені американським математиком Уільямом Карушем і іншими у кінці 1940-х років, стали важливим інструментом для розпізнавання рішень і управління поведінкою алгоритмів. Важливий ранній алгоритм рішення нелінійних програм дав норвезький економіст, лауреат Нобелівської премії Рагнар Фриш в середині 1950-х років. Цікаво, що його підхід втратив популярність впродовж декількох десятиліть, і знову з'явився як життєздатний і конкурентний підхід тільки в 1990-х роках.

Інші важливі алгоритмічні підходи включають послідовне квадратичне програмування, в якому вирішується наближене завдання з квадратичною метою і лінійними обмеженнями для отримання кожного кроку пошуку; і штрафні методи, у тому числі "метод множників ", в якому точки, які не задовольняють обмеженням, спричиняють за собою штрафні санкції в цілях відвертання відвідування алгоритмами алгоритмів.

Лауреат Нобелівської премії американський економіст Гаррі М. Марковиц дав поштовх для нелінійної оптимізації в 1958 році, коли він сформулював проблему пошуку ефективного інвестиційного портфеля як завдання нелінійної оптимізації з квадратичною цільовою функцією. Методи нелінійної оптимізації нині широко використовуються у фінансах, економіці, виробництві, управлінні, моделюванні погоди і в усіх галузях техніки.

Теорія

Завдання оптимізації є нелінійним, якщо цільова функція $f(x)$ або будь-яке з обмежень нерівності $c\ i\ (x) \ \leq 0,\ i = 1,\ 2,.,\ m$ або обмежень рівності $d_j(x) = 0, j = 1, 2,., n,$ - нелінійні функції вектору змінних x . Наприклад, якщо x містить компоненти x_1 і x_2, то функція $3 - 2\ x_1 - 7x_2$ є лінійною, тоді як функції $(x_1)^3 + 2\ x_2$ і $3\ x_1 + 2\ x_1\ x_2 + x_2$ є нелінійними.

Нелінійні проблеми виникають, коли мета або обмеження не можуть бути виражені у вигляді лінійних функцій без збитку для якої-небудь істотної нелінійної особливості системи реального світу.

Наприклад, згорнута конформація білкової молекули, як вважають, зводить до мінімуму деяку нелінійну функцію відстаней між ядрами її складових атомів, а самі ці відстані є нелінійними функціями положень ядер. У фінансах ризик, пов'язаний з портфелем інвестицій, вимірюється як відхилення дохідності портфеля є нелінійною функцією сум, вкладених в кожен цінний папір в портфелі.

У хімії концентрація кожної хімічної речовини в розчині часто є нелінійною функцією часу, оскільки реакції між хімічними речовинами зазвичай відбуваються по експоненціальних формулах.

Нелінійні завдання можна класифікувати за декількома властивостями. Існують проблеми, в яких мета і обмеження є гладкими функціями, і існують нерівні проблеми, в яких нахил або значення функції можуть різко змінитися. Існують необмежені проблеми, в яких мета полягає в тому, щоб мінімізувати (чи максимізувати) цільову функцію $f(x)$ без обмежень на значення x, і існують обмежені проблеми, в яких компоненти x повинні задовольняти певним межам або іншим складніші взаємовідносини. У опуклі завдання, графік цільової функції і здійснима безліч обоє опуклий (де множина опукла, якщо в нім міститься пряма, що сполучає будь-які дві точки великої кількості).

Ще один особливий випадок квадратичне програмування, в якому обмеження лінійні, а цільова функція квадратична; тобто він містить члени, кратні твору двох компонентів x. (Наприклад, функція $3(x_1)^2 + 1{,}4\, x_1 x_2 + 2(x_2)^2$ є квадратичною функцією від x_1 і x_2.)

Ще один корисний спосіб класифікації нелінійних завдань - по числу змінні (тобто компоненти x). Грубо кажучи, проблема називається "великою", якщо в ній більше тисячі змінних, хоча поріг "великості" постійно збільшується, оскільки комп'ютери стають усе більш потужними.

Інша корисна відмінність - між завданнями, які обчислювально "дороги" для оцінки, і тими, які відносно дешеві, як у разі лінійного програмування. Алгоритми нелінійного програмування зазвичай виходять з послідовності припущень змінного вектору x (відомого як ітерації і розрізняного по верхніх індексах x_1, x_2, x_3,.) з метою зрештою визначити оптимальне значення x .

Часто непрактично визначати глобально оптимальне значення x. У цих випадках треба погодитися з локальним оптимумом - кращим значенням в деякому регіоні з можливих рішень. Кожна ітерація вибирається на основі знань про обмеження і цільові функції, зібрані на більше ранніх ітераціях. Більшість алгоритмів нелінійного програмування націлена на певний підклас проблем.

Наприклад, деякі алгоритми спеціально призначені для великих гладких необмежених завдань, в яких матриця других похідних функції $f(x)$ містить декілька ненульових елементів і є дорогою для оцінки, тоді як інші алгоритми спеціально призначені для завдань опуклого квадратичного програмування і т.д.

Програмне забезпечення для вирішення завдань оптимізації доступно як комерційно, так і у відкритому доступі. На додаток до програм комп'ютерної оптимізації доступні декілька мов моделювання оптимізації, які дозволяють користувачеві описати проблему на інтуїтивно зрозумілих термінах, які потім автоматично переводяться в математичну форму, потрібну програмним забезпеченням для оптимізації.

Графічний метод

Графічний метод рішення задач лінійного програмування використовується, коли є тільки дві змінні рішення. Якщо в завданні три або більше змінних, графічний метод не підходить. В цьому випадку ми використовуємо симплекс-метод, який обговорюється в наступному розділі.

Ми розпочнемо з того, що дамо декілька важливих визначень і понять, які використовуються в методах рішення задач лінійного програмування.

1. **Рішення**. Набір значень змінних рішення, що задовольняють усім обмеженням лінійного про-проблема граматики називається вирішенням цієї проблеми.
2. **Можливе рішення.** Будь-яке рішення, яке також задовольняє обмеженням позитивності, проблема називається можливим рішенням.
3. **Оптимальне здійсниме рішення.** Будь-яке здійсниме рішення, яке максимізувало або мінімізує мету Функція називається оптимальним здійснимим рішенням.
4. **Можлива область**. Загальна область, визначувана усіма обмеженнями і позитивністю обмеження LPP називається здійснимою областю.
5. **Кутова точка**. Кутова точка можливого регіону - це точка в можливому регіоні, яка перетин двох граничних ліній.

Наступна теорема є основною теоремою лінійного програмування.

Теорема 1.1 Якщо оптимальне значення цільової функції в завданні лінійного програмування існує, то це значення повинне зустрічатися в одній (чи більше) з кутових точок допустимої області.

Щоб вирішити завдання лінійного програмування з двома вирішальними змінними, використовуючи графічний метод, ми використовуємо алгоритм описаний нижче.

Графічний метод рішення ЗПП:

Крок 1. Сформулюйте завдання лінійного програмування.

Крок 2. Графік можливої області і знайти кутові точки. Координати кутових точок можна отримати за допомогою перевірки або шляхом рішення двох рівнянь ліній, що перетинаються в цій точці.

Крок 3. Складіть таблицю з вказівкою значення цільової функції в кожній кутовій точці.

Крок 4. Визначте оптимальне рішення з таблиці в кроці 3. Якщо проблема має тип максимізації (мінімізації), розв'язок відповідає найбільшому (найменшому) значенню цільової функції, є оптимальним рішенням ЗЛП.

Тепер ми використовуватимемо цю процедуру для вирішення деяких ЗЛП, де модель вже була визначена.

Ми використовуємо **Теорема 1.1** для ілюстрації. Графік ЗЛП показаний на Рисунку 1.2.

Крок 2. Межа допустимої області складається з ліній, отриманих зі зміни нерівностей в рівність; тобто лінії:

$2S+4T=12$ і $3S+2T=10$

Крок 3. Кутові точки (чи крайні точки) і цільові функціональні значення, що відповідають їм:

Екстремальний прибуток $(P=S+1.5T)$

(0, 0)	0
$(\frac{10}{3}, 0)$	$\frac{10}{3}$
(2, 2)	5
(0, 3)	4.5

Крок 4. Тому ми робимо висновок, що оптимальним рішенням є $S = 2$, $T = 2$, що відповідає прибутку $P = 5$. Таким чином, прибуток максимізувався, коли 2 одиниці стандартної якості і 2 одиниці фарби вищої якості були виготовлені.

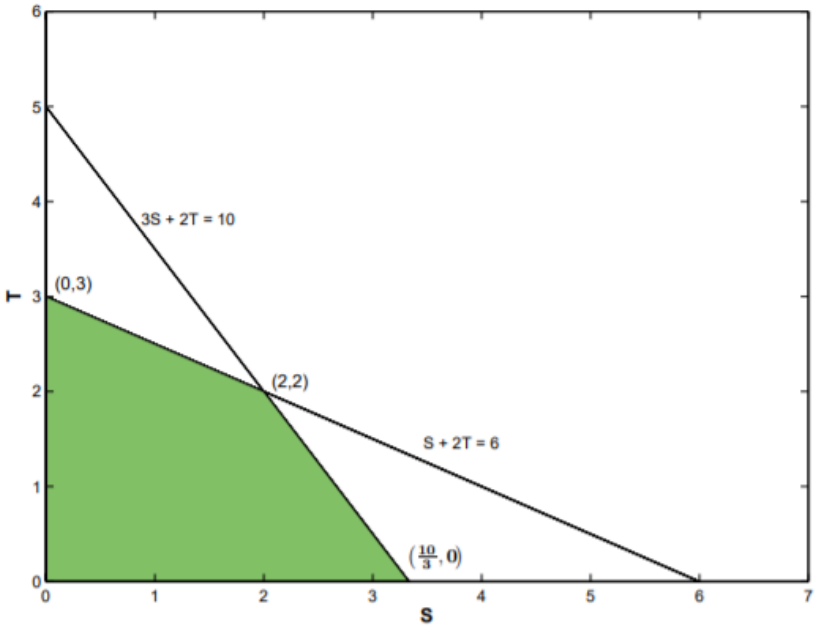

Рисунок 1.2 - Графічне рішення моделі прикладу прототипу

Симплекс метод

Графічний метод рішення може бути використаний лише для вирішення лінійних задач, визначених з використанням тільки дві або три змінні. Неможливо вирішити лінійні моделі, де більше трьох змінних за допомогою графічного рішення, а, отже, необхідно використати процедуру алгебри. У 1949 році Джордж Б. Данциг розробив симплекс метод рішення завдань лінійного програмування.

Симплекс-метод призначений для застосування тільки після того, як лінійна модель виражається в такій формі: Стандартна форма.

Кажуть, що лінійна модель знаходиться в стандартній формі, якщо всі обмеження рівні, і кожне з значень в векторі b і всіх змінних в моделі не негативні.

$$\max(\min)\ z = c^T x$$

при умові

$$Ax = b$$

$$x \geq 0$$

Якщо мета полягає в максимізації, то говорять, що модель знаходиться в максимізації стандартна форма. Інакше модель називається стандартною мінімізацією.

Модель маніпуляції

Лінійні моделі мають бути написані в стандартній формі, щоб вирішити за допомогою симлекс метод. Простими маніпуляціями будь-яка лінійна модель може бути перетворена в еквівалентний написаний в стандартній формі. Цільова функція, обмеження і змінні можуть бути перетворені в міру необхідності.

Цільова функція

Обчислення мінімального значення функції z еквівалентно обчисленню максимальне значення функції – z

$$\min z = \sum_{j=1}^{n} c_j x_j \Longleftrightarrow \max (-z) = \sum_{j=1}^{n} -c_j x_j$$

Наприклад, $\min z = 3x_1 - 5x_2$ і $\max (-z) = -3x_1 + 5x_2$ еквівалент; значення змінних x_1 і x_2, які роблять значення z мінімальним і - z максимум однакові. Він тримає $\min z = -\max (-z)$.

Обмеження

а) обмеження лінійної моделі можна помножити на - 1, наприклад так:

$$\sum_{j=1}^{n} a_{ij} x_j \geq b_i \quad \Longleftrightarrow \quad \sum_{j=1}^{n} -a_{ij} x_j \leq -b_i$$

Наприклад, якщо ми помножимо обмеження нерівності $2x_1 + 3x_2 \leq -2$ по – 1, отримуємо еквівалентне обмеження нерівності $-2x_1 - 3x_2 \geq 2$.

(б) Нерівність може бути перетворена в рівність.

$$\sum_{j=1}^{n} a_{ij}\, x_j \geq b_i \quad \Longleftrightarrow \quad \sum_{j=1}^{n} a_{ij}\, x_j + y = b_i$$
$$\sum_{j=1}^{n} a_{ij}\, x_j \leq b_i \quad \Longleftrightarrow \quad \sum_{j=1}^{n} a_{ij}\, x_j - y = b_i$$

Змінна *y*, використана в попередніх перетвореннях, називається *слабкою змінною*. У обох випадках передбачається, що $y \geq 0$.

Наприклад, обмеження $x_1 - 4x_2 \leq 4$ і $x_1 - 4x_2 + y = 4$ еквівалентно, якщо виконане $y \geq 0$.

Обмеження рівності можна перетворити на два обмеження нерівності, діючи так:

$$\sum_{j=1}^{n} a_{ij}\, x_j = b_i \quad \Longleftrightarrow \quad \sum_{j=1}^{n} a_{ij}\, x_j \leq b_i \quad \text{і} \quad \sum_{j=1}^{n} a_{ij}\, x_j \geq b_i$$

Наприклад, обмеження рівності $-2x_1 + x_2 = 2$ що еквівалентно мають обидва наступні обмеження нерівності : $-2x_1 + x_2 \leq 2$ і $-2x_1 + x_2 \geq 2$.

Змінні

Змінні мають бути ненегативними. Це перетворення, які можуть бути використані у разі, якщо вони не:

- Якщо змінна x_j визначена як негативна, $x_j \leq 0$, то підстановка $x_j = -x_j'$ робить ненегативну змінну x_j'
- Якщо змінна x_j не має обмежень по знаку, її можна замінити на $x_j = x_j' - x_j''$, де $x_j' \geq 0$ і $x_j'' \geq 0$.
  - Якщо $x_j' > x_j''$, то отримуємо $x_j > 0$.
  - Якщо $x_j' < x_j''$, то отримуємо $x_j < 0$.
  - Якщо $x_j' = x_j''$, то отримуємо $x_j = 0$.

Приклад

Беручи до уваги еквівалентності, представлені раніше, давайте перетворимо наступну лінійну модель в стандартну форму максимізації:

$$\min z = x_1 + 2x_2 + x_3$$

при умові $x_1 + x_2 - x_3 \geq 2$

$$x_1 - 2x_2 + 5x_3 \leq -1$$

$$x_1 + x_2 + x_3 = 4$$

$x_1 \geq 0$, $x_2 \leq 0$, x_3 : необмежено

- Об'єктивна функція

Ми перетворимо цільову функцію у функцію максимізації:

$$\max (-z) = -x_1 - 2x_2 - x_3$$

- Обмеження

Перше обмеження нерівності може бути записане як обмеження рівності віднімаючи не негативну змінну слабкого місця x_4,

$$x_1 + x_2 - x_3 - x_4 = 2.$$

Додаємо змінну слабкого місця x_5 до другого обмеження,

$$x_1 - 2x_2 + 5x_3 + x_5 = -1.$$

Ми множимо на - 1 обмеження, так що значення в правій частині стає позитивні,

$$-x_1 + 2x_2 - 5x_3 - x_5 = 1.$$

Третє обмеження правильно написане: $x_1 + x_2 + x_3 = 4$.

- Змінні

Змінна x_1 визначена не негативний. Змінна x_2 не визначена не негативною,

$x_2 \leq 0$, тому ми визначаємо змінну x_2' як $x_2' = -x_2$ і зміните знак коефіцієнти змінної x_2 в моделі при заміні її на x_2'. Змінна x_3 необмежена в знаку. Тому ми визначаємо $x_3 = x_3' - x_3''$ і замінити його в моделі.

Отримана лінійна модель знаходиться в стандартній формі максимізації і може бути записана у виді:

$$\max(-z) = -x_1 + 2x_2' - x_3' + x_3'' + 0x_4 + 0x_5$$

$$\text{при умові} \quad x_1 - x_2' - x_3' + x_3'' - x_4 = 2$$

$$-x_1 - 2x_2' - 5x_3' + 5x_3'' - x_5 = 1$$

$$x_1 - x_2' + x_3' - x_3'' = 4$$

$$x_1, x_2', x_3', x_3'', x_4, x_5 \geq 0$$

Додавання або віднімання слабких змінних не повинні змінити мету функція. Таким чином, їх коефіцієнт в цільовій функції дорівнює нулю.

Генетичний алгоритм

Генетичний алгоритм - це метод рішення як обмежених, так і необмежених завдань оптимізації, грунтований на природному відборі, процесі, який стимулює біологічну еволюцію.

Генетичний алгоритм неодноразово модифікує сукупність індивідуальних рішень. На кожному етапі генетичний алгоритм випадковим чином вибирає окремих людей з поточної популяції в якості батьків і використовує їх для виробництва дітей для наступного покоління. Упродовж подальших поколінь населення "еволюціонує" у напрямі оптимального рішення. Ви можете застосовувати генетичний алгоритм для вирішення різних завдань оптимізації, які не зовсім підходять для стандартних алгоритмів оптимізації, включаючи завдання, в яких цільова функція є переривчастою, такою, що не диференціюється, стохастичною або сильно нелінійною. Генетичний алгоритм може вирішувати проблеми змішане цілочисельне програмування, де деякі компоненти обмежені цілочисельними значеннями.

Генетичний алгоритм використовує три основні типи правил на кожному кроці для створення наступного покоління з поточної популяції:

- Правила відбору відбирають людей, що називаються батьками, які вносять вклад в населення наступного покоління.

- Правила кросовера об'єднують двох батьків, щоб сформувати дітей для наступного покоління.
- Правила мутації застосовують випадкові зміни до окремих батьків, щоб сформувати дітей.

Нехай K = кількість поколінь і $P_k = \{x^i : x^i \in R^n, i = 1 ..., I_k\}$ безліч геномів в k-м поколінні, де I_k означає розмір для $k = 1 ..., K$.

Псевдокод для генетичного алгоритму тепер можна записати таким чином:

Алгоритм "Генетичний оптимізатор"

Створіть початкову популяцію $P_1 = \{x^i \in R^n, i = 1 ..., I_1\}$.

Для $k = 1,2 ..., K$

Якщо знайдено досить хороше рішення $x \in P_k$,

то $x^* = x$, вихід

ще

Виконати операцію кросовера для ряду геномів популяції P_k . Це призводить до збільшення P_k .

Виконайте операцію мутації на геномах, які були створені під час кросовера. Виберіть набір кращих геномів від популяції P_k до популяції P_{k+1} для наступного покоління.

кінець Якщо

кінець Для.

Визначення чисельності населення $I_1,..., I_k$ залишається відкритим в цьому загальному описі.

1.2 Метод Монте-Карло та його застосування

Поняття Монте-Карло потрапляє в розділ експериментальної математики. У звичайній математиці виведення виводяться з постулатів (дедукція). У експериментальній математиці висновки робляться із спостережень (індукції). Методи Монте-Карло складають ту гілку експериментальної математики, яка займається експериментами з випадковими подіями (головним чином: випадковими числами). Методи Монте-Карло можуть бути імовірнісного або детермінованого типу.

Зазвичай першою згадкою про метод Монте-Карло є знаменитий експеримент з голкою графа де Буона (1733), французького біолога (1707-1788), Рис. 1.3, де він вказав, що якщо голку завдовжки *L* кинути на площину з паралельними лініями на відстані *D* один від одного *(D > L),* то вона має вірогідність $P = \frac{2L}{\pi D}$ впасти так, що перетинає одну з ліній.

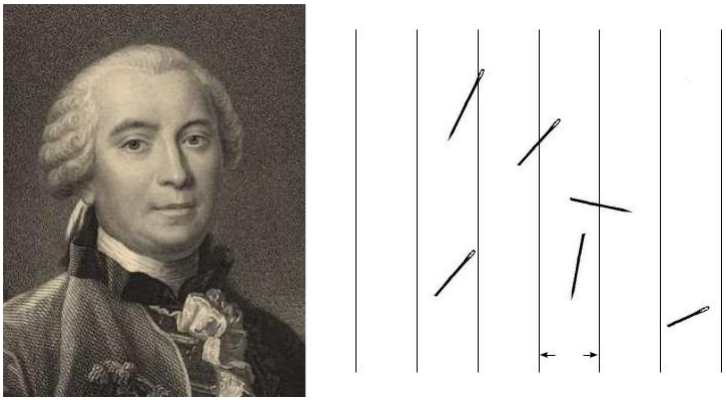

Рисунок 1.3 - Яка вірогідність p, що голка (довжина L), яка випадково падає на лист, перетинає одну з ліній (відстань D)

Пізніше також Лаплас запропонував цю процедуру визначити шляхом підрахунку π числа хрестиків *n* в *N* повтореннях експерименту. Потім:

$$\frac{n}{N} = \frac{2L}{\pi D} \rightarrow \pi \approx \frac{2L}{D} \cdot \frac{N}{n}$$

Це історичне використання процедури "Монте-Карло" має усі ключові особливості методу:

- Конвергенція: біля N=100000 проб потрібно тільки для 2 цифр після коми. Зближення відбувається повільно, але надійно.
- Прозорість: метод інтуїтивно зрозумілий, навіть без яких-небудь математичних міркувань.
- оцінки помилок, оптимізація : оцінки помилок і оптимальний вибір L; D забезпечуються теорією вірогідності (біномний розподіл, статистична дисперсія як 2-й Центральний момент і т. д.).

Сучасне використання методів Монте-Карло в епоху цифрових комп'ютерів було ініційоване новаторською роботою Джона фон Неймана і Станіслава Улама в області розробки термоядерної зброї. Їм також приписують те, що вони вигадали Фразу "Монте-Карло".

Багато монографій по методах Монте-Карло розпочинаються з введення в теорію вимірів і, зокрема, в елементарну теорію вірогідності. Хоча ми також введемо і використовуватимемо належний математичний словник, ми будемо, відносно чисто математичних аспектів, посилатися на них і значною мірою покладатися на інтуїтивне значення.

Ми посилаємося, зокрема, на класичну монографію Хаммерсли і Хэндскомба. Ця книга є коротким і дуже читаним оглядом Монте-Карло. Відмітно, що теоретичні основи сьогодні залишаються дуже 4 Детлев Рейтер так само, як і в 1964 році, коли ця книга була уперше опублікована. Просто додатки сьогодні набагато складніші. Ілюстративні приклади інтеграції по методу Монте-Карло і деякі передові методи в справжньому введенні будуть грунтовані на цьому тексті.

Базовий принцип

Принцип полягає в тому, щоб знайти (оцінити) середні значення, т.е. математичні очікування, *I* з яких деякі компоненти системи. Якщо необхідно вирішити детерміністичне завдання (напр. певний інтеграл, матрична інверсія...) спочатку треба винайти стохастик система *(Ω, σ, p, X)* така, що середнє значення (= математичне очікування *E (X)*) цього система співпадає з шуканим рішенням *I* детермінованого завдання. У інших випадках, якщо завдання визначене вже в імовірнісних термінах, то це може бути безпосередньо узята за основу для "аналогового" моделювання і оцінки.

У будь-якому випадку: *I* - це єдина числова величина, що представляє інтерес (а не ціле число функціональна залежність), і завжди можна було б думати про я як про деяке визначене складений.

Прості інтуїтивні інтерпретації дані нижче, але в абстрактному виді математичні терміни ця стохастична модель задається імовірнісним простором *(Ω, σ, p, X). Ω* - множество елементарних (випадкових) подій ω, σ-поле – множина підмножини Ω, яким вимірна функція p привласнює значення ("вірогідність") з інтервалу [0, 1], таке, що аксіоми Колмогорова для а "вірогідність " виконується. *X* - це випадкова величина на *Ω*, призначення (зазвичай дійсне) число (чи вектор) для кожної випадкової події, наприклад: *X (ω) → R*, таке, що *I = E (X),* очікуване значення X.

Математичне очікування *E (X)* і дисперсія $\sigma^2 (X)$ визначаються як перший момент і другий центральний момент відповідно, і, якщо не вказане інше
Отже, ми припускаємо, що вони обоє існують:

$$E(X) := \int_{\Omega} dp\ X \text{ і } \sigma^2(X) := \int_{\Omega} dp\ (X - E(X))^2$$

(Примітка: $E(X) = E_p(X)$, $\sigma^2(X) = \sigma^2{}_p(X)$, т. е. моменти *X* звичайно залежать від імовірнісної міри *p*.)

Стохастичне наближення до *I* тоді виходить шляхом отримання незалежної послідовності випадкових подій $\omega = [w_i, i = 1 \ldots N]$ відповідно до вірогідності закон *p* і оцінка

$$E(X_N) = I_N(\omega) = \frac{1}{N}\sum_{i=1}^{N} X(\omega_i)$$

"Оцінювач " I_N - це усього лише середнє арифметичне багатьох *(N)* оцінок *X* (ω_i) випадкового експерименту, що повторюється.

Навіть без якої-небудь з цих абстрактних математичних передумов інтуїтивно ясно, що I_N сходитиметься, в деякому розумінні, до *E (X),* отже, до *I* по конструкції, оскільки число зразків *N* збільшується.

Проте "закони великих чисел" і "центральні граничні теореми" теорії вірогідності не лише дають переконливі математичні докази того, що це Монте Процедура Карло є точною ("неупередженою"), але і те, що вона сходиться : $I_N \to I$ для $N \to \infty$, хоча і повільно (з $1/\sqrt{\sqrt{N}}$). Зокрема, "центральна гранична теорема теорії вірогідності стверджує, що розподіл випадкової величини в $I_N(\omega)$, для великих досить *N*, сходиться до гауса розподілу, з середнім значенням $I = E(X)$ і дисперсією $\sigma^2(I_N) = \sigma^2(X) / N$. Звідси типові результати статистичних досліджень аналіз помилок відповідно до законів гаусів розподілу застосовний, наприклад, також до отриманих результатів. Таким чином, в додатках Монте-Карло поширеною практикою є цитування результатів у виді

$$I \approx I_N \pm \sigma(I_N) \quad або \quad I \approx I_N \pm 2 \cdot \sigma(I_N)$$

Які мають довірчі рівні біля 66% і 95% відповідно.

Звичайно, в додатках дисперсія $\sigma^2(X)$ зазвичай ще важче вичислити, чим середнє значення *E (X).* Тому воно замінюється на наступне: "емпірична дисперсія":

$$s^2 = \frac{1}{N-1}\sum_{i=1}^{N}[X(\omega_i) - E(X_N)]^2 = \frac{1}{N-1}\left(\sum_{i=1}^{N} x^2(\omega_i) - \frac{1}{N}\left[\sum_{i=1}^{N} x(\omega_i)\right]^2\right)$$

і один з них також, згідно із зробленими припущеннями, для великого розміру вибірки *N*

$$s^2 \to \sigma^2 \quad і \quad \sigma^2(I_N) \approx s^2{}_N = \frac{1}{N} s^2$$

Отже, для досить великих N в оцінках гаусів погрішності σ можна безпечно замінити на s_N, принаймні для великого розміру вибірки*: $N \geq 100$.* У протилежний випадок: $N \leq 100$

Генерація випадкових чисел

Метод Монте-Карло грунтований на нашій здатності робити випадкові числа беруться з будь-якого конкретного розподілу вірогідності *F (x),* або, якщо воно існує, з функція щільності розподілу вірогідності:

$$\text{f (X) з F (x)} = \int_{-\infty}^{x} dt\, f(t)$$

Прикладами можуть служити змочування поверхні дощем (рівномірний розподіл), радіоактивний розпад (розподіл Пуассона) або розподіл швидкостей молекул у Газі (розподіл гауса). У загальному випадку: закон вірогідності має бути відомо або з теорії, або з експерименту, або з правдоподібності.

Теорема з теорії заходів станів:

Теорема 1.2 Кожна (імовірнісна) міра μ на R може бути розкладена на зважена сума $\mu = p_1\, \mu c + p_2\, \mu d + p_3\, \mu a$ з трьох частин:

Частина 1 має безперервний розподіл (з щільністю вірогідності f (x));

Частина 2 має дискретний розподіл;

Частина 3 – це патологічний ("сингулярний по відношенню до μ") вклад.

Можливість генерувати випадкові числа дуже корисна в деяких видах програм, зокрема в іграх, програмах наукового або статистичного моделювання. Візьмемо, приміром, ігри без рандомных (чи ще "випадкових") подій - монстри завжди атакуватимуть вас однаково, ви завжди знаходитимете одні і ті ж предмети/артефакти, макети темниць і підземель ніколи не мінятимуться і так далі, загалом, сюжет такої гри не дуже і навряд чи ви довго протримаєтеся.

Генератор псевдовипадкових чисел

Оскільки ж генерувати випадкові числа? У реальному житті ми часто кидаємо монету (орел/решка), кістки або перетасовуємо карти. Ці події включають так багато фізичних змінних (наприклад, сила тяжіння, тертя, опір повітря і так далі), що вони стають майже неможливими для прогнозування/контролю і видають результати, які в усіх сенсах є випадковими.

Проте комп'ютери не призначені для використання фізичних змінних - вони не можуть кинути монету, кістки або перетасувати реальні карти. Комп'ютери живуть у контрольованому електричному світі, де є тільки два значення (true і false), чогось середнього між ними немає. За своєю природою комп'ютери призначені для отримання прогнозованих результатів. Коли ми говоримо комп'ютеру порахувати, скільки буде 2 + 2, ми завжди хочемо, щоб відповіддю було 4 (не 3 і не 5).

Отже, комп'ютери нездатні генерувати випадкові числа. Замість цього вони можуть імітувати випадковість, що досягається за допомогою генераторів псевдовипадкових чисел.

Генератор псевдовипадкових чисел - це програма, яка набуває стартового/початкового значення і виконує з ним певні математичні операції, щоб конвертувати його в інше число, яке зовсім не пов'язане із стартовим. Потім програма використовує нове згенероване значення і виконує з ним ті ж математичні операції, що і з початковим числом, щоб конвертувати його ще в одно нове число - третє, яке не пов'язане ні з першим, ні з другим. Застосовуючи цей алгоритм до останнього згенерованого значення, програма може генерувати цілий ряд нових чисел, які здаватимуться випадковими (за умови, що алгоритм буде досить складним). Мови C і C++ мають свої власні вбудовані генератори випадкових чисел. Вони реалізований в двох окремих функціях, які знаходяться в заголовному файлі cstdlib:

Функція srand () встановлює передаване користувачем значення в якості стартового. srand () слід викликати тільки один раз: на початку програми (зазвичай у верхній частині функції main()).

Функція rand () генерує наступне випадкове число в послідовності. Воно знаходитиметься в діапазоні від 0 до RAND _ MAX (константа в cstdlib, значенням якої є 32 767).

Однорідні випадкові числа

Однорідні випадкові числа є основою для генерації випадкових чисел з усіма іншими законами розподілу. Випадкова величина розподіляється рівномірно на інтервалі [a, b], якщо щільність розподілу *f* рівна:

$$f(x) = 1/(b-a)\, x_{[a,b]},\ x \in R$$

з $x\ [a, b] = 1$, Якщо x знаходиться в інтервалі $[a, b]$ і $f(x) = 0$ у іншому місці. Класичний метод генерації однорідних випадкових чисел на $[0, 1]$ полягає в наступному, так звані "лінійні конгруентні генератори випадкових чисел", які визначаються за допомогою рекурсії:

$$\varepsilon_{n-1} = [a\ \varepsilon_n + b]$$

Тут "чарівне" множене, м якщо часто вибирається найбільше ціле число уявний на машині ($m = 2^{32}$ і т. д.), і b має бути простим до m. Докази для конкретного вибору (великих) параметрів a і m, що генератор досягає максимально можливого періоду $m - 1$ різні випадкові числа являються досить громіздко. Оптимальні параметри вибору зазвичай знаходять експериментально. Кінцева періодичність обмежує точність тільки в дуже великих випадках розрахунки, наприклад, на сучасних масивно-паралельних обчислювальних системах. А швидше за все тонким питанням є також "незалежність" усієї послідовності випадкових чисел.

Неоднорідні випадкові числа

Як було сказано вище нам треба розглядати тільки дискретні розподіли і безперервні розподіли.

Знаходження випадкових чисел із заданим дискретним розподілом тривіальне:

Дискретний розподіл з k елементарними результатами, поміченими природними числа *{0, 1, 2,. . ., k},* може бути задано з допомогою:

$$P\ (X = i) = p_i \geq 0,\ \sum_{i=0}^{k} p_i = 1 \text{ і } F\ (i) = P\ (X \leq i) = \sum_{j=0}^{i} p_j$$

при *P (X = i)* вірогідність події *i*, *F* є (кумулятивним) розподілом.

Нехай ξ-однородное випадкове число на *[0, 1],* тоді випадкова величина *X* з: *X = i*, Якщо *F (i - 1) < ξ ≤ F (i)* розподіляється відповідно до *F.*

Методи Монте-Карло можна розглядати як сукупність обчислювальних методів для (зазвичай наближеного) рішення математичних задач, в яких використовуються фундаментальні випадкові вибірки. В рамках цієї системи найчастіше вирішуються два класи статистичних проблем: інтеграція і оптимізація. Ця стаття концентрується на першому: це (приблизний) розрахунок інтегралів з використанням наборів випадкових вибірок, про які зазвичай думають люди, коли посилаються на метод Монте-Карло. Методологія Монте-Карло також широко використовується при моделюванні фізичних, хімічних і біологічних систем.

В галузі освіти методи Монте-Карло найбільш цікаві як обчислювальний пристрій для виконання статистичного висновку. Багато цікавих моделі мають надзвичайно складну структуру і не можуть бути легко вирішені з використанням традиційних методів. В рамках байєсівської парадигми вся інформація, на якій може грунтуватися висновок, кодується в межах апостеріорного розподілу ймовірностей. Використовуючи методи Монте-Карло, ми можемо охарактеризувати ці розподілу і розрахувати очікування по ним: основний метод логічного висновку.

Приклад. Методи Монте-Карло перевертають звичайну проблему статистики: замість того, щоб оцінювати випадкові величини детермінованим способом, випадкові величини використовуються для оцінки детермінованих величин. Наприклад, один простий експеримент Монте-Карло розглядає дощ, який випадає рівномірно випадковим чином (тобто місцезнаходження краплі дощу може бути інтерпретовано, як реалізація рівномірно розподіленої випадкової величини) в деякій квадратної області простору, і коло, вписаний в цей квадрат, без посилання на будь-яку формальну теорію ймовірностей, інтуїтивно зрозуміло, що ймовірність падіння краплі дощу в будь-якій області в межах квадрата повинна бути пропорційна площі цієї області і не залежати від її місця розташування. Отже, ймовірність, РТО, що крапля дощу лежить всередині вписаного кола, може бути виражено через їх області. Якщо квадрат має боку довжини *2r*, коло повинен мати радіус r і

Само по собі це може здатися не особливо цікавим. Однак, висловивши π як функцію цієї ймовірності, її оцінки можна використовувати для апроксимації π. Аналітична робота неможлива: отримання ймовірності вимагає знання π. Інтуїтивно можна оцінити цю ймовірність, порахувавши частку крапель дощу, які лежать всередині кола: якщо спостерігаються n крапель дощу і m з них лежать всередині кола, то можна оцінити p, використовуючи, На Рисунку 1.4 показано комп'ютерне моделювання, в якому 500 крапель дощу були рівномірно розподілені по квадрату - в цьому випадку і, визначаючи через відносини між p і π ,, Погана оцінка π, враховуючи використовувані обчислювальні зусилля, але, тим не менш, це оцінка. Насправді *m* є реалізацією біноміальної *(n, p)* випадкової величини.

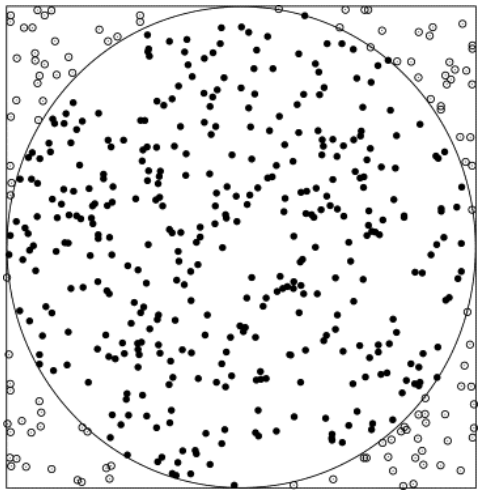

Рисунок 1.4 - Використання методу Монте-Карло для пошуку числа π

Переваги методу Монте-Карло

Моделювання по методу Монте-Карло дає ряд переваг в порівнянні з детерміністичним або "одноточечним аналізом" аналізом:

- Імовірнісні результати. Результати показують не лише те, що може статися, але і наскільки вірогідний кожен результат.
- Графічні результати. Із-за даних, які генерує симуляція Монте-Карло, легко створювати графіки різних результатів і їх вірогідності появи. Це важливо для передачі результатів іншим зацікавленим сторонам.
- Аналіз чутливості. Тільки у декількох випадках детерміністичний аналіз утрудняє визначення того, які змінні впливають на результат найбільш. У моделюванні Монте-Карло легко побачити, які вхідні дані зробили найбільший вплив на підсумкові результати.
- Аналіз сценарію. У детермінованих моделях дуже важко змоделювати різні комбінації значень для різних вхідних даних, щоб побачити результати дійсно різних сценаріїв. Використовуючи моделювання по методу Монте-Карло, аналітики можуть точно визначити, які вхідні дані мали які значення разом досягши певних результатів. Це неоцінимо для подальшого аналізу.

- Співвідношення входів. У симуляції Монте-Карло можна моделювати взаємозалежні стосунки між вхідними змінними. Для точності важливо уявити, як, насправді, коли одні чинники підвищуються, інші відповідно підвищуються або знижуються.

1.3 Постановка задачі

Проаналізувавши проблеми, що описуються моделями лінійного та нелінійного програмування та переваги методу Монте-Карло по рішенню цих проблем, поставимо наступне завдання дослідження:

1) Адаптувати метод Монте-Карло до рішення задач лінійного та нелінійного програмування;
2) Створити методику рішення задач багатокритеріальних;
3) Провести тестові розрахунки та зробити висновки по застосовності методу Монте-Карло.

2 АДАПТАЦІЯ МЕТОДУ МОНТЕ-КАРЛО ДЛЯ РІШЕННЯ ПРОБЛЕМИ

2.1 Математична постановка задачі

Завданням нелінійного програмування (завданням НП) називається завдання знаходження максимуму (мінімуму) нелінійної функції багатьох змінних, коли на змінні є (немає) обмеження типу рівності або нерівностей.

У стандартній формі завдання нелінійного програмування (завдання НП) можна записати в наступному виді:

$$max\ f\ (x_1, x_2, \ldots, x_n) \qquad (2.1)$$

при обмеженнях

$$g_i\ (x_1, x_2, \ldots, x_n) \leq 0,\ i = \overline{1, m}$$

Якщо ввести позначення

$$X = (x_1, \ldots, x_n)\ \in R^n$$

то завдання *(2.1)* НП набере простішого вигляду:

$$max\ f\ (X) \qquad (2.2)$$

при обмеженнях

$$g_i\ (X) \geq 0,\ i = \overline{1, m}$$

Множина M = $\{X|g_i\ (X) \geq 0,\ i = \overline{1, m}\} \subset R^n$ - називатимемо безліччю допустимих рішень ЗНП. Зрозуміло, що завдання *(2.2)* можна переписати таким чином:

$$\max_{X \in M} f\ (X)$$

- Допустиме рішення $X^* \in M$ називається оптимальним рішенням задачі НП, якщо

$$\max_{X \in M} f\ (X) = f\ (X^*)$$

Помітимо, що останній запис означає, що

$$f\ (X^*) \geq f\ (X),\ \text{для будь якого } X \in M$$

тобто $f\ (X^*)$ найбільше значення функції f на множині M.

- Число $f(X^*)$ називатимемо значенням завдання НП.
- Надалі припускатимемо, що функції f : $R^n \to R^1$ і $g_i : R^n \to R^1$, $i = \overline{1,m}$

- дифференцируемы в області визначення.

2.2 Алгоритм рішення проблеми

Метод Монте-Карло складається з чотирьох етапів:

1. Побудова математичної моделі системи, що описує залежність модельованих характеристик від значень стохастичних змінних.
2. Встановлення розподілу вірогідності для стохастичних змінних.
3. Встановлення інтервалу випадкових чисел для кожної стохастичної змінної і генерація випадкових чисел.
4. Імітація поведінки системи шляхом проведення багатьох випробувань і отримання оцінки модельованої характеристики системи при фіксованих значеннях параметрів управління. Оцінка точності результату.

Перший етап. Стохастична імітаційна модель (ІМ) деякої реальної системи може бути представлена як динамічна система, яка під впливом зовнішніх випадкових вхідних сигналів (вхідних змінних) змінює свій стан (випадкові змінні стани), що у свою чергу призводить до зміни вихідних сигналів (вихідних змінних):

$\mathbf{S}_i\mathbf{+1} = \mathbf{F}\ (\mathbf{S}_i,\mathbf{I}_{i+1})$

$\mathbf{U}_i = \mathbf{R}(\mathbf{S}_i)$

де **F**, **R** - вектор-функції;

$\mathbf{I}_i$, $\mathbf{U}_i$, $\mathbf{S}_i$ - вектори відповідно вхідних, вихідних змінних і змінних стану системи в тактовий момент моделювання *i*.

Другий етап. Випадкові величини, використовувані в ЇМ, можуть бути дискретними або безперервними. У першому випадку необхідно знати їх розподіли, в другому - щільність розподілів. Ці залежності можуть бути відомі з теорії, визначені в результаті спеціальних досліджень або задані в якості гіпотези. Точність моделі (за інших рівних умов) залежить від того, наскільки точно задані вказані розподіли (щільність розподілів).

Третій етап. Моделювання випадкових величин при комп'ютерних імітаційних експериментах робиться за допомогою датчика псевдовипадкових чисел, передбаченого у будь-якій сучасній мові програмування. Звичайно це датчик випадкових чисел з рівномірним розподілом на інтервалі [0, 1]. Якщо відома вірогідність настання подій, то, використовуючи такий датчик, можна відповідати на питання: "Яке з N можливих подій сталося?" чи "Якого значення набула випадкова величина?"

Припустимо, що в ІМ використовується випадкова величина X, що приймає дискретні значення $x_1, x_2..., x_N$ з вірогідністю відповідно $p_1, p_2..., p_N$ ($\sum_{k=1}^{N} \mathrm{p} = 1$). Отримання деякій реалізації цієї змінної в моделі робиться таким чином. Будується функція розподілу випадкової величини X. Вказана функція визначається за допомогою рівності $F(X) = \sum p_k$, в якому підсумовування поширюється на усі індекси, для яких $x_k < X$. За допомогою датчика випадкових чисел отримують випадкове число і з відрізку [0, 1].

З рівномірності розподілу отримуваних випадкових чисел виходить, що вірогідність отримання випадкового числа з довільного інтервалу, включеного в [0, 1], дорівнює довжині цього інтервалу. Тому вірогідність реалізації $X = x_k$ дорівнює вірогідності gпадання ориманого від датчика випадкового числа u в довільний інтервал завдовжки p_k на відрізку [0, 1]. Можна, таким чином, стверджувати, що якщо чергове число і датчика задовольняє нерівностям $0 < u \le p_1$, то має місце реалізація $X = x_1$, у разі

$p_1 < i \le p_1 + p_2$ - реалізація $X = x_2$ і так далі. У загальному випадку для $k = 2 ..., N$: якщо $\sum_{j=1}^{k-1} p_i < u \le \sum_{j=1}^{k} p_j$, то $X = x_k$.

Помітимо, що межі вказаних нерівностей співпадають зі значеннями побудованої вище функції розподілу *F (X).*

Зручніше, проте, мати справу не з дробовими значеннями меж інтервалів, в які попадає випадкове число *i*, а з їх цілочисельними значеннями, тим паче, що за допомогою датчиків випадкових чисел можна генерувати числа з будь-якого діапазону. Щоб отримати цілі значення меж інтервалів, досить помножити усе p_k на 10^d, где d - ціле мінімальне значення якого дорівнює максимальній точності (максимальному числу знаків після десяткової точки) чисел p_k, $k = 1,..., N$. Наприклад, якщо $\{p_k\} = \{0,3; 0,153; 0,5; 0,047\}$, те мінімальне значення d дорівнює 3 (усе p_k треба помножити на 1000). Таким чином, 10^d визначає довжину інтервалу значень даної випадкової величини в ІМ.

Четвертий етап. Точність статистичних оцінок параметрів реальної системи залежить від числа спостережень (об'єму вибірки). Погрішності в оцінках обумовлені як статистичним характером самої моделі, так і впливом початкових даних (початкового стану імітаційної системи), а також можливою автокореляцією послідовних значень деякого параметра в процесі моделювання. Очевидно, що зі збільшенням числа випробувань точність моделювання повинна зростати. Зважаючи на те що збільшення об'єму вибірки пов'язане із зростанням витрат на моделювання, важливо уміти визначати мінімальне число випробувань, необхідне для досягнення заданої точності оцінки із заданою вірогідністю.

Широке поширення отримали два методи статистичних випробувань. Один з них припускає проведення досить великого числа *T* послідовних спостережень в течія одного прогону моделі (одного сеансу імітування).
Інший метод полягає в реалізації *m* незалежних прогонів моделі, тобто в *m*- кратному повторенні одного і того ж циклу імітування.

При цьому, якщо ми хочемо отримати в сумі *T* спостережень, впродовж кожного прогону можна робити по *T/m* (допустимий, що це число ціле) спостережень. Обидва методи дають приблизно однаковий результат.

Нехай значення y_t (t = 1..., T) є результатами T послідовних вимірів значень випадкової величини у під час одного і того ж сеансу імітації. Середнє за часом значення у визначається вираженням

$$\overline{y} = \sum_{t=1}^{T} y_t / T.$$

Позначимо через μ математичне очікування випадкової величини *y*. Тоді для досить великого *T* отримуємо

$$\overline{y} \approx \mu.$$

Оцінка дисперсії у (якщо часовий ряд не є автокорельованим) має вигляд,

$$\mathbf{D}(\overline{y}) = \mathbf{D}(y)/T,$$

де $D(y)$ - дисперсія випадкової величини *y*.

Для оцінки якості результатів, отриманих методом Монте-Карло при невідомій дисперсії спостережуваної випадкової величини, припустимо, що *Z* - характеристика, яка має бути визначена (вірогідність події, математичне очікування, дисперсія і тому подібне), а ξ - її значення, що уточнюється у міру накопичення даних, залишається випадковим внаслідок обмеженості числа *T* проведених спостережень. У цих умовах можна говорити про вірогідність $p(|Z - \xi| < \varepsilon)$ по відношенню до характеристики, що цікавить нас. Величина $|Z - \varepsilon|$ є погрішність в оцінці Z, а ε - деяка допустима її межа.

З нерівності Чебишева виходить:

$$p(|Z - \xi| < \varepsilon) \geq 1 - \mathbf{D}_\xi(T)/\varepsilon^2.$$

З цієї нерівності слідує

$$\mathbf{D}_\xi(T) < (1 - p)\,\varepsilon^2,$$

звідки при заданих *p* і ε і при відомій залежності $\boldsymbol{D}_\varepsilon$ (Т) можна знайти гранично необхідне *T*.

Відомо, що істинна дисперсія вибіркового розподілу для розрахункового середнього обернено пропорційна до сумарного числа спостережень T, тобто

$$\mathbf{D}_{\xi}(T) = d/T,$$

де d не залежить від T.

На початку імітаційного процесу необхідне число спостережень визначити зазвичай не вдається, оскільки d невідоме. Тому, як правило, експеримент проводять в два етапи.

На першому етапі число випробувань вибирається відносно невеликим, в результаті визначається величина d. Після цього можна вже визначити, скільки додаткових спостережень потрібні, щоб була досягнута необхідна точність.

Граничне число спостережень T_0 визначається формулою

$$T_0 = d/[(1 - p)\varepsilon^2].$$

При будь-якому числі спостережень більше T_0 забезпечується необхідна точність.

3 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ТА ТЕСТОВІ РОЗРАХУНКИ

3.1 Комп'ютерна реалізація алгоритмів та основні її складові

Отримання випадкових значень типу double :

```
double random(double min, double max){
    return (double)(rand())/RAND_MAX*(max - min) + min;
 }
```

Перевірка отриманих результатів та за необхідності збереження їх :

```
if (pow(x,2)+pow(y,2)<=25 && x*y>=4){
                f = 3*x + 4*y;
                if(f >= f_max){
                    max_x = x;
                    max_y = y;
                    f_max = f;
                }
  }
```

Виконуючи цю послідовність дій певну кількість разів :

```
for(int i = 0 ; i<n ; i++)
```

3.2 Тестові розрахунки

За умовою завдання треба знайти максимальне значення функції $F = 3x + 4y$

при $\begin{cases} x^2 + y^2 \leq 25 \\ y \geq \frac{4}{x} \\ x \geq 0 \\ y \geq 0 \end{cases}$

Розв'яжемо це завдання математичним шляхом:

Побудуємо область допустимих рішень. Перша нерівність - дуга окружності з центром на початку координат і з радіусом R = 5. Друге нерівність геометрично представляє гіперболу виду $y = \frac{4}{x}$.

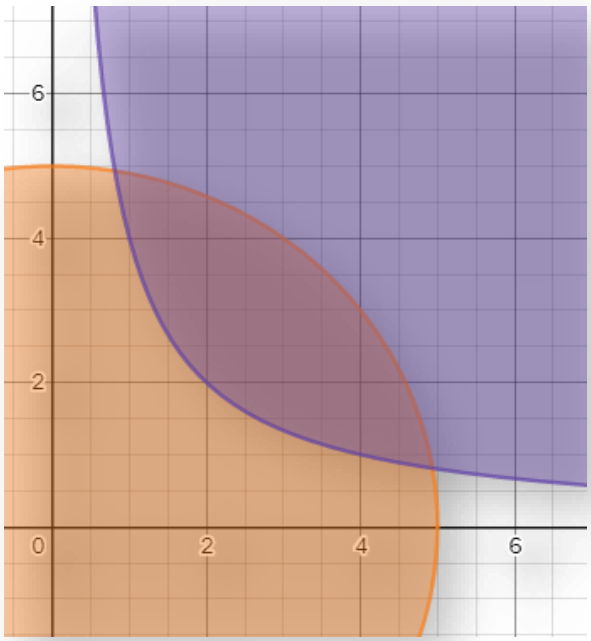

Рисунок 3.1 - Область допустимих рішень задачі

Лінія рівня - це пряма, що має рівняння $y = -\frac{3}{4}x$. Рішення завдання отримаємо, якщо лінію рівня будемо паралельно самій собі переміщати в напрямку ОДР. Там де лінія рівня вийде з ОДР, тобто в точці дотику з іншого обмежувальної лінією, буде знаходитися рішення на максимум функції цілі.

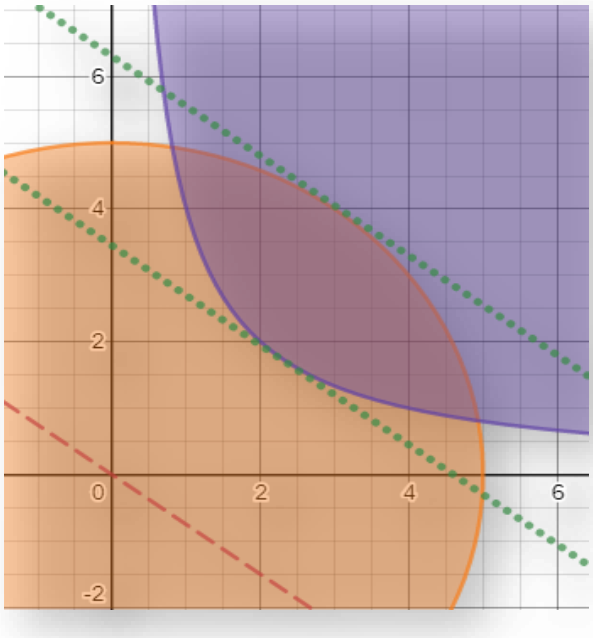

Рисунок 3.2 – Лінія рівня

Для знаходження продифференціюємо рівняння кола і отримаємо:$2x + 2yy' = 0$, звідки

$$y' = -\frac{x}{y}$$

Так як це кутовий коефіцієнт дотичної до окружності, то рівність:

$$-\frac{x}{y} = -\frac{3}{4}$$

Отримуємо рішення: при $x = 3$, $y = 4$, отримуємо $F_{max} = 25$.

Якщо порівняти відповіді , то можна побачити, що програма працює правильно.

x = 3.0922 (x = 3)

y = 3.92841 (y = 4)

$F_{max} =$ 24.9903 ($F_{max} = 25$)

Порівняльний аналіз

Порівнювати будемо кількість взятих випадкових величин для розв'язку поставленої задачі.

1. N = 10

```
Кількість випробувань : 10
max :
При x: 2.60436 y: 2.76145 F = 18.8589
```

2. N = 100

```
Кількість випробувань : 100
max :
При x: 2.17043 y: 4.37306 F = 24.0035
```

3. N = 1000

```
Кількість випробувань : 1000
max :
При x: 2.46471 y: 4.35007 F = 24.7944
```

4. N = 5000

```
Кількість випробувань : 5000
max :
При x: 2.98212 y: 3.99961 F = 24.9448
```

5. N = 10000

```
Кількість випробувань : 10000
max :
При x: 2.97687 y: 4.00985 F = 24.97
```

6. N = 50000

```
Кількість випробувань : 50000
max :
При x: 3.09222 y: 3.92841 F = 24.9903
```

Якщо порівнювати кожний випадок , то можна побачити , що N = 10 – занадто маленька кількість точок , а вже N = 50000 – добра кількість для точного розв’язку задачі.

Асимптотична оцінка

Асимптотичну оцінку алгоритму, який розглядається визначити недосить складно. Вона залежить від кількості точок, які потравляють в обмежену область дії програми. Також оскільки програма не містить складних циклів, можна зробити висновок, що асимптотичною оцінкою даного алгоритму для визначення максимального значення функції є $O(n)$.

Приклад застосування алгоритму

Виходячи з вище викладеного можна побудувати загальний алгоритм рішення багатокритерійної задачі, алгоритм у вигляді блок-схеми, показаний на Рисунку 3.3

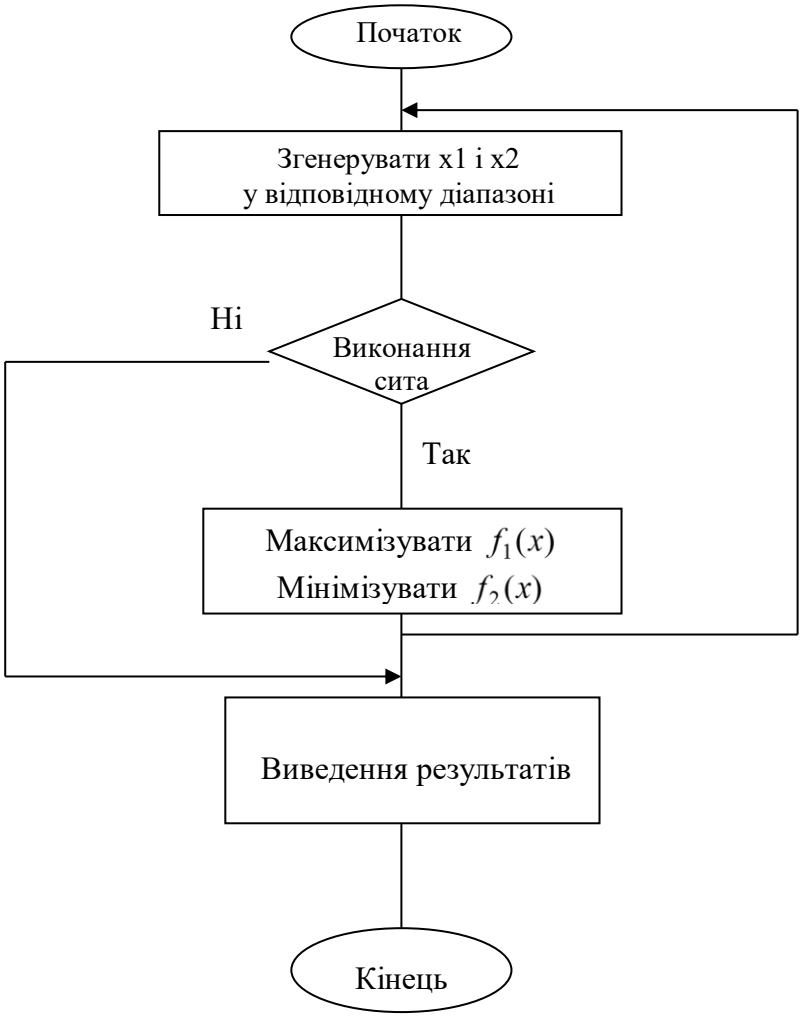

Рисунок 3.3 - Структурная схема алгоритма

На Рисунку 3.3 показаний алгоритм, грунтований на генеруванні багатовимірних векторів для перевірки "ситом". Залежно від отриманого результату при перевірки ситом робиться відповідний висновок. При цьому перевіряється перевірка на максимальність і мінімальність функцій тих, що беруть участь в рішенні багатокритерійної задачі.

Основними блоками блок схеми тут являється : по-перше, блок вибору області допустимих рішень задачі - це ті варійовані змінні, які задовольняють системі обмежень; по-друге, блок одночасної максимізації - мінімізації усіх критеріїв.

Особливість полягає в наступному, що точки оптимимума кожного критерію окремо не співпадають і тому слід шукати якийсь компроміс в рішенні, щоб задовольнити виконання усіх критеріїв одночасно.

Одним з відомих методів пошуку рішень є метод пошуку Парето - ефективних рішень. Покажемо його дію на прикладі.

Нехай є безліч варіантів рішення. По кожному з варіантів визначені значення усіх критеріїв. Представимо безліч оцінок варіантів рішення в просторі критеріїв (Рисунок 3.4).

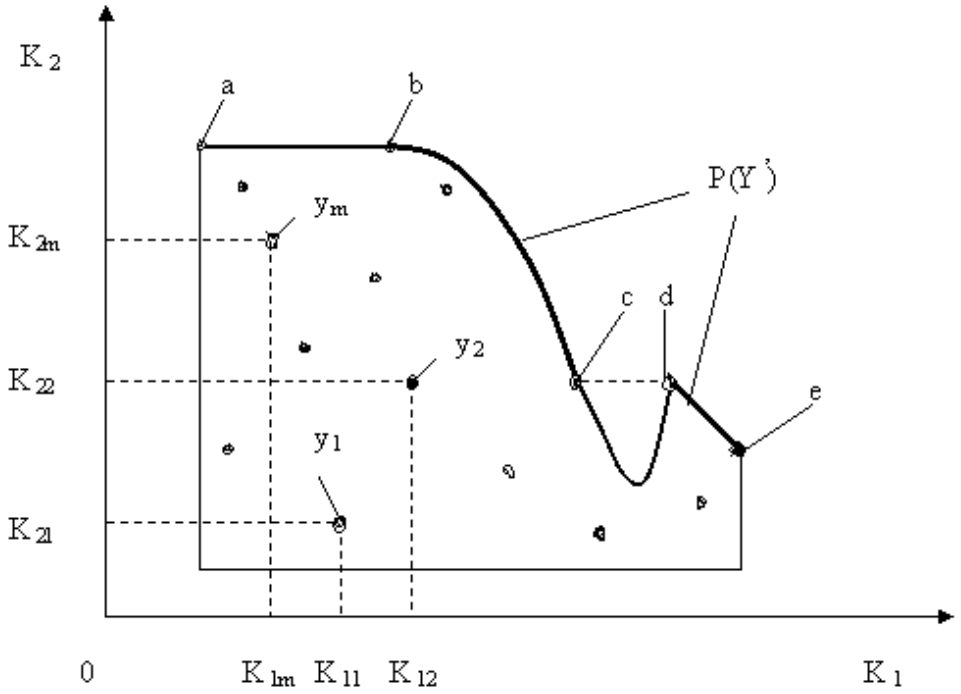

Рисунок 3.4 - Ілюстрація пошуку Парето - ефективних рішень

На рисунку прийняті наступні позначення:

- K_1 і K^2 - критерії оцінки варіантів рішення;
- $Y = \{y_1, y_2 ., y_m\}$- безліч оцінок альтернативних варіантів рішення;
- K_{11}, K_{12} ., K_{1m} - значення першого критерію для 1, 2, ., m - го варіанту рішення;
- , K_{22} ., K_{2m} - значення другого критерію для 1, 2, ., m - го варіанту рішення;
- P (Y) - безліч Парето - ефективних оцінок рішень.

Правило. Безліч Парето - ефективних оцінок P (Y') є "північно - східну" межу безлічі Y без тих його частин, які паралельні одній з координатних осей або лежать в "глибоких" провалах .

Для випадку, зображеного на Рисунку 3.4, Парето - ефективні оцінки складаються з точок кривої (bc), виключаючи точку (c), і лінії (de).

Переваги методу : 1) Критерії рівнозначні; 2) Метод математично об'єктивний.

Недолік методу : 1) Одно остаточне рішення виходить тільки в окремому випадку, тобто кількість Парето - ефективних рішень, як правило, більше за одне.

Мультиплікативний критерій.

Цільова функція тут записується таким чином:

$$F(X) = \prod_{i=1}^{n} CiFi(X) \rightarrow \max(\min),$$

де П - знак твору; c_i - ваговий коефіцієнт i- го приватного критерію;

f_i - числове значення i- го приватного критерію.

Переваги мультиплікативного критерію :

- Не потрібно нормування приватних критеріїв.
- Практично завжди визначається одно оптимальне рішення.

Недоліки:

- Труднощі (суб'єктивізм) у визначенні вагових коефіцієнтів.
- Перемножування різної розмірності.
- Взаємна компенсація значень приватних критеріїв.
- В якості тестового прикладу, проведемо рішення наступної задачі : при обмеженнях

$$\begin{cases} \delta_1 : x_1 + 2x_2 \geq 4; \\ \delta_2 : 3x_1 + x_2 \geq 7; \\ \delta_3 : -3x_1 + 5x_2 \leq 17; \\ \delta_4 : 5x_1 - x_2 \leq 23; \\ \delta_5 : 3x_1 - 4x_2 \leq 7; \\ x_1, x_2 \geq 0 \end{cases}$$

Необхідно максимізувати функцію $f_1(x) = x_1 + 4x_2$

І мінімізувати функцію $f_2(x) = 3x_1 - x_2$

Вирішуючи графічно цю задачу, отримаємо наступні графік (Рисунок 3.5).

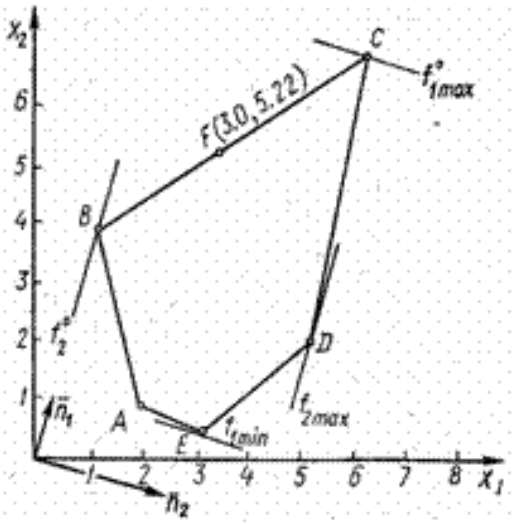

Рисунок 3.5 - Графічне рішення тестової задачі

Графічно оптимальне рішення цієї задачі для виконання одного критерію, наприклад за критерієм $f_1(x)$ отримати неважко. Воно знаходиться в точці C $(x_1 = 6; x_2 = 7)$.

Йому відповідає $f_1 = f(x_1 = 6; x_2 = 7) = 34$.

Мінімальне значення буде в точці E $(x_1 = 3; x_2 = 0{,}5)$, $f_{1min} = 5$. За другим критерієм $f_2(x)$ оптимальне рішення буде в точці B(1,4) йому відповідає $f_2^0 = f_2(1{,}4) = -1$, а найгірше значення критерію $f_2(x)$ - в точці $D(5,2)$: $f_{2\min} = 13$. Якщо розглядати завдання з позицій виконання одночасно двох критеріїв оптимальності, то відрізок BC є ефективним планом. Т.е. питання полягає у виборі точки на відрізку, одночасно точкою оптимуму, що являється, за двома критеріями.

Якщо розглянути, що ці критерії являються рівнозначними по важливості, то такою точкою буде точка по середині відрізку BC. Наприклад, на малюнку показана точка F з координатами: F(3.0; 5.22) в якій значення функцій мети рівні відповідно: $f_1 = 23.88$; $f_2 = 3.78$.

Реалізація цього прикладу на ЕОМ чисельним методом Монте-Карло дає значення функцій мети, представлених в таблиці 3.1.

Аналіз отриманих результатів

В результаті виконання програми отримано рішення, яке відповідає графічному рішенню поставленої задачі (Таблиця 3.1)

x_1	1.001	5.985	3.025
x_2	4.000	7.002	5.215
f_1	17.001	33.993	23.885
f_2	-0.997	10.863	3.86

Таблиця 3.1 - Результат виконання програми

В результаті проведеного експерименту, ми бачимо, що результати обчислень співпали з допустимою точністю. Так само проведений аналіз поведінки цільової функції від кількості точок (Рисунок 3.6), що генеруються.

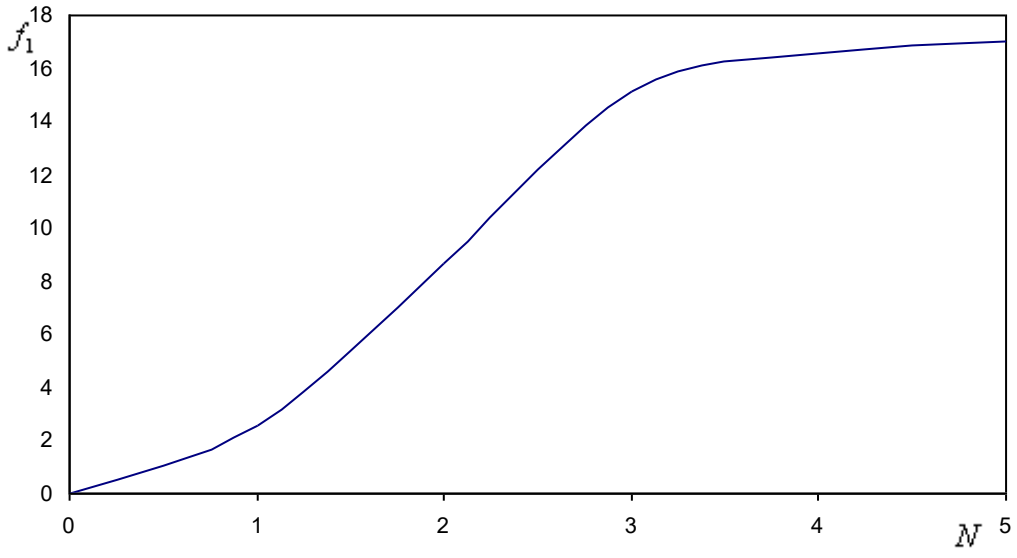

Рисунок 3.6 - Залежність цільової функції f_1 від N.

На Рисунку 3.6 шкала N вимірюється в одиницях рівних 10^6. Аналізуючи Рисунок 3.6 видно, що максимальне значення критерійної функції досягає 99,6% від еталонного і що в межі модель рішення побудована на методі Монте-Карло дає хороший збіг з точним рішенням.

ВИСНОВКИ

У даній роботі було розв’язано задачу на знаходження максимального значення для заданої функції за допомогою методу Монте–Карло, тобто побудовано алгоритм для вирішення задачі нелінійного програмування.

Переваги алгоритму: проста реалізація, метод дозволяє досягти необхідної точності результатів.

Недоліки алгоритму: точність рішень залежить від кількості N, які можуть бути виконані (цей недолік стає менш значущим зі збільшенням швидкодії комп'ютера), високі вимоги до обчислювальної потужності і значні витрати часу на проведення розрахунків.

Результати тестового прикладу свідчать про те, що програмна реалізація алгоритму працює правильно. Отже, завдання виконано в повному обсязі.

СПИСОК ЛІТЕРАТУРИ

1. Математичні методи дослідження операцій : підручник / Є. А. Лавров, Л. П. Перхун, В. В. Шендрик та ін. – Суми : Сумський державний університет, 2017. – 212 с.
2. Таха Хэмди А. Исследование операций/ Таха Хэмди А. – Москва: Вильямс, 2016. – 912 с.
3. Mazhdrakov M. The Monte Carlo Method: Engineering Applications/ Mazhdrakov M. , Benov D. , Valkanov N. – ACMO Academic Press, 2018. – 250 p.
4. *Bruns P.* Monte-Carlo Tree Search in the game of Tantrix: Cosc490 Final Report (PDF) *(Report), 2017. – 15 p.* [Електронний ресурс] – Режим доступу до ресурсу: https://www.tantrix.com/Tantrix/TRobot/MCTS%20Final%20Report.pdf
5. Lemire D. Fast Random Integer Generation in an Interval, 2019. - 15 p. [Електронний ресурс] – Режим доступу до ресурсу: https://arxiv.org/pdf/1805.10941.pdf
6. Fudenberg Drew, Maskin Eric. The folk theorem in repeated games with discounting or with incomplete information // A Long-Run Collaboration On Long-Run Games. — World Scientific, 2009. — P. 209–230.
7. Krauth W. Introduction to Monte Carlo algorithms, 2006. – 41 p. [Електронний ресурс] – Режим доступу до ресурсу: https://cel.archives-ouvertes.fr/cel-00092936/document

ДОДАТОК

Код програми

```
#include <iostream>
#include <math.h>
#include <time.h>
using namespace std;

double random (double min, double max){
  return (double) (rand ()) / RAND_MAX * (max - min) + min;
}

class Point{
  //координати точки
  double x;
  double y;
  public:
  void fun (int n){
      double f;
      double f_max = 0;
      double max_x = 0, max_y = 0;
      int k = 0;
      for (int i = 0; i < n; i++){
        x = random (0, 5);
        y = random (0, 5);
        // перевірка входження точки в задану область
        if (pow (x, 2) + pow (y, 2) <= 25 && x * y >= 4){
            f = 3 * x + 4 * y;
            // перевірка на макимум
            if(f >= f_max){
                // запам'ятовування максимальних значень
                max_x=x;
                max_y = y;
```

```
                f_max = f;
            }
        }
      }
    cout << "max : " << endl;
    cout << "При x: " << max_x << " y: " << max_y << " F = " <<f_max
<<endl;
  }
};


int main (){
    setlocale(LC_ALL,"ukr");
    srand ((unsigned int) time (0));
    cout << "Кількість випробувань : ";
    int n;
    cin >> n;
    Point z;
    z.fun(n);
    system ("pause");
    return 0;
}
```