Міністерство освіти і науки України
Університет митної справи та фінансів

Факультет інноваційних технологій
Кафедра комп'ютерних наук та інженерії програмного забезпечення

Кваліфікаційна робота магістра

на тему: «Розробка оптимізаційних алгоритмів для вирішення економічних задач»

Виконав: студент групи К23-1М

Спеціальність 122 Комп'ютерні науки

Андрасович М.В.
(прізвище та ініціали)

Керівник к.е.н. Яковенко Т. Ю.
(науковий ступінь, вчене звання, прізвище та ініціали)

Рецензент Дніпровський державний технічний університет
(місце роботи)

доцент кафедри математичного моделювання та системного аналізу
(посада)

к.т.н., доц. Волосова Н.М.
(науковий ступінь, вчене звання, прізвище та ініціали)

Дніпро – 2025

АНОТАЦІЯ

Андрасович М.В. Розробка оптимізаційних алгоритмів для вирішення економічних задач.

Дипломна робота на здобуття освітнього ступеня магістр за спеціальністю 122 «Комп'ютерні науки» – Університет митної справи та фінансів, Дніпро, 2025.

У кваліфікаційній роботі досліджено сучасні алгоритми оптимізації в економіці, їхні можливості, обмеження та практичне застосування для вирішення актуальних економічних задач. Дослідження підтвердило, що використання оптимізаційних моделей дозволяє значно підвищити ефективність управлінських рішень, мінімізувати витрати та оптимально розподілити ресурси навіть в умовах обмежень та невизначеності.

Робота висвітлює переваги сучасних методів, зокрема їхню здатність адаптуватися до швидкозмінних умов економічного середовища, обробляти великі обсяги даних і враховувати багатофакторні залежності. Лінійні та нелінійні методи програмування розглянуто як основні інструменти для вирішення задач управління виробничими процесами, логістикою, фінансами та плануванням. Еволюційні алгоритми демонструють високу ефективність у розв'язанні задач оптимізації з нелінійними залежностями.

На основі проведеного аналізу було розроблено рекомендації щодо вибору оптимальних алгоритмів залежно від специфіки задачі, доступності даних та обмежень. Практична цінність роботи полягає у можливості застосування розроблених моделей і підходів у різних сферах економіки для підвищення ефективності виробництва, управління ресурсами, зменшення витрат та покращення стратегічного планування.

Ключові слова: оптимізація, економічна ефективність, алгоритми, машинне навчання, еволюційні методи, теорія ігор, лінійне програмування, оптимізаційні моделі.

ABSTRACT

Andrasovych M.V. Development of optimization algorithms for solving economic problems.

Diploma thesis for obtaining a master's degree in specialty 122 «Computer Science» – University of Customs and Finance, Dnipro, 2025.

The master's thesis investigates modern optimization algorithms in economics, their capabilities, limitations and practical application for solving current economic problems. The study confirmed that the use of optimization models can significantly increase the efficiency of management decisions, minimize costs and optimally allocate resources even under conditions of constraints and uncertainty.

The paper highlights the advantages of modern methods, including their ability to adapt to rapidly changing economic conditions, process large amounts of data, and take into account multifactorial dependencies. Linear and nonlinear programming methods are considered as the main tools for solving problems of production process management, logistics, finance, and planning. Evolutionary algorithms demonstrate high efficiency in solving optimization problems with nonlinear dependencies.

Based on the analysis, recommendations for choosing the optimal algorithms depending on the specifics of the problem, data availability, and constraints were developed. The practical value of the work lies in the possibility of applying the developed models and approaches in various sectors of the economy to improve production efficiency, resource management, cost reduction, and strategic planning.

Keywords: optimization, economic efficiency, algorithms, machine learning, evolutionary methods, game theory, linear programming, optimization models.

ЗМІСТ

ВСТУП

Оптимізація є однією з найважливіших задач, що постають перед сучасною економічною наукою, адже вона охоплює широкий спектр процесів, які визначають ефективність використання обмежених ресурсів у різних сферах економіки. Використання математичних моделей та алгоритмів для оптимізації економічних процесів дозволяє значно підвищити ефективність управлінських рішень, що є надзвичайно актуальним у часи глобалізації, високої конкуренції та невизначеності ринкових умов. Оскільки сучасна економіка характеризується величезною кількістю змінних, складністю взаємозв'язків між ними, а також постійними змінами у зовнішньому середовищі, пошук нових та більш ефективних методів оптимізації набуває все більшої важливості.

Актуальність теми кваліфікаційної роботи обумовлена необхідністю вдосконалення існуючих методів оптимізації для забезпечення сталого розвитку та підвищення конкурентоспроможності економічних систем на різних рівнях. У сучасних умовах економічні агентства та організації стикаються з необхідністю прийняття рішень, які мають враховувати безліч факторів, від національних економічних політик до глобальних трендів, таких як цифровізація, автоматизація та інтеграція у міжнародні торговельні системи. Тому важливість дослідження новітніх підходів та алгоритмів для ефективного вирішення економічних задач є не лише теоретичною, а й практичною проблемою.

Метою цієї роботи є аналіз сучасних алгоритмів оптимізації, їх застосування в різних сферах економіки, а також розробка рекомендацій для підвищення ефективності економічних процесів за допомогою впровадження новітніх методів оптимізації. Окрім цього, передбачається дослідження ефективності різних підходів до оптимізації та їх застосування у специфічних

економічних ситуаціях, що дозволить зробити більш обґрунтовані висновки щодо вибору найбільш ефективних методів у залежності від конкретних умов.

Для досягнення поставленої мети передбачається вирішення низки завдань:

– аналіз сучасного стану теорії та практики оптимізації в економіці;

– вивчення найбільш ефективних алгоритмів оптимізації, що застосовуються в економічних процесах;

– розробка та впровадження моделей оптимізації для розв’язання конкретних економічних задач;

– порівняння ефективності різних алгоритмів оптимізації в контексті їх практичного застосування в різних галузях економіки;

– оцінка впливу оптимізаційних рішень на економічні показники та ефективність господарських суб’єктів.

Об’єктом дослідження є процеси оптимізації, які використовуються для покращення ефективності економічних рішень. Особливу увагу буде приділено алгоритмам, що застосовуються для управління виробничими процесами, ресурсами, логістикою та фінансами, а також методам, що використовуються для стратегічного планування та прогнозування економічних змін.

Предметом дослідження є сучасні алгоритми оптимізації, що застосовуються для розв’язання різноманітних економічних задач. Ці алгоритми включають, зокрема, методи лінійного та нелінійного програмування, генетичні алгоритми, алгоритми на основі теорії еволюції, методи машинного навчання та інші новітні підходи, які мають потенціал для широкого застосування у вирішенні складних економічних завдань.

Методологічною основою дослідження є аналіз та синтез теоретичних підходів до оптимізації економічних процесів, математичні методи та моделювання, а також використання кількісних та якісних методів аналізу для оцінки ефективності застосованих алгоритмів. У роботі будуть використані

методи комп’ютерного моделювання, аналізу даних, статистичні методи для порівняння результатів різних оптимізаційних підходів, а також методи емпіричного дослідження для перевірки результатів на практиці.

Практична значимість цієї роботи полягає в тому, що розроблені алгоритми оптимізації та рекомендації можуть бути безпосередньо використані економічними агентами для покращення управління ресурсами, підвищення ефективності виробництва, зменшення витрат та максимізації прибутку. Окрім того, робота може бути корисною для розробки нових інструментів у сфері фінансових технологій, управлінських стратегій та інноваційних технологій, що є важливими для розвитку як окремих підприємств, так і національних економік в цілому.

Наукова новизна роботи полягає в дослідженні та інтеграції сучасних методів оптимізації з різних галузей науки та технологій в економічну практику. Окрему увагу буде приділено новітнім алгоритмам, що ще не знайшли широкого застосування в економічних дослідженнях, а також порівнянню їх ефективності з традиційними методами. Розроблені підходи та рекомендації для оптимізації економічних процесів дозволять значно покращити точність економічних прогнозів та підвищити рівень ефективності використання ресурсів у господарських системах.

Таким чином, актуальність, мета та завдання дослідження підкреслюють важливість аналізу сучасних методів оптимізації для ефективного управління економічними процесами, що дозволяє зробити значний внесок у розвиток економічної науки та практики.

Структура кваліфікаційної роботи. Кваліфікаційна робота складається з трьох розділів. Обсяг кваліфікаційної роботи – 93 сторінки. Робота містить 10 рисунків. Перелік використаних джерел налічує 16 посилань.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ ДОСЛІДЖЕННЯ

1.1 Визначення та сутність оптимізації в економіці

Оптимізація в економіці є одним з основних напрямків сучасної економічної теорії та практики. Це процес, що полягає в пошуку найкращих варіантів або рішень для досягнення певних цілей за допомогою розподілу обмежених ресурсів. В основі оптимізації лежить концепція досягнення максимального результату при мінімальних витратах або досягнення певного рівня результату при мінімізації витрат [1]. Оптимізація застосовується у різних сферах економіки, таких як фінанси, виробництво, управління підприємствами, планування і багато інших. Визначення оптимізації в економічному контексті залежить від мети, яку ставить перед собою економічний агент, будь то окрема особа, підприємство або урядова організація. Економічна оптимізація включає не тільки математичні методи, а й стратегічне планування, аналітичний підхід до управління ресурсами. Вона застосовується на всіх рівнях економічної діяльності – від мікроекономіки до макроекономіки, охоплюючи такі поняття, як оптимальний розподіл ресурсів, максимізація прибутку, мінімізація витрат або максимізація добробуту. При цьому оптимізація в економіці передбачає врахування численних факторів, що обмежують можливості економічних агентів: обмеженість ресурсів, конкуренція, державне регулювання та інші.

Основна сутність оптимізації полягає в досягненні такого стану економічної системи, при якому економічні агенті (фірми, домогосподарства, уряди) реалізують свої цілі з максимальним ефектом. Для цього застосовуються різні методи, серед яких найбільш поширеними є лінійне програмування, нелінійне програмування, динамічне програмування, теорія ігор, а також різноманітні числові методи, які дозволяють розв’язувати складні

економічні задачі. У той час як класична економічна теорія зосереджена на досягненні рівноваги через ринкові механізми, оптимізація дозволяє не лише знайти рівноважні стани, а й шукати шляхи для більш ефективного використання наявних ресурсів. У галузі мікроекономіки оптимізація є важливим інструментом для досягнення максимального прибутку або мінімізації витрат у межах підприємства [2]. Наприклад, оптимізація виробничого процесу може полягати у визначенні найефективнішої комбінації праці та капіталу, що дозволяє отримати максимальний обсяг продукції при найменших витратах. Інший приклад оптимізації в мікроекономіці – це максимізація споживацької корисності, де споживач прагне оптимально розподілити свої доходи між різними товарами та послугами, щоб отримати найбільшу можливу задоволеність своїх потреб.

Водночас, в макроекономіці оптимізація має більш широкий контекст, зокрема, у питаннях планування національних ресурсів, бюджетного дефіциту, рівня інфляції та безробіття. У цьому контексті оптимізація полягає в досягненні збалансованого економічного розвитку, що передбачає ефективне використання ресурсів на рівні держави. Це може включати оптимізацію податкової політики, соціальних програм, інвестиційних стратегій та інших аспектів державного управління. У зв’язку з глобалізацією економіки, важливими стають також питання міжнародної оптимізації, зокрема, щодо оптимального розподілу виробничих процесів між різними країнами, що дозволяє досягти економічних переваг завдяки порівняльним перевагам. Оптимізація в економіці не є одноразовим процесом, це постійно змінювана та динамічна система, що вимагає регулярних коригувань у залежності від змін зовнішнього середовища. Наприклад, зміни в законодавстві, технологіях, демографічних тенденціях, політичних факторах можуть значно впливати на ефективність економічних рішень. Тому оптимізація вимагає постійного моніторингу та коригування стратегій в реальному часі. Це стає особливо важливим у сучасному світі, де швидкі зміни

та непередбачуваність можуть значно впливати на фінансову стабільність як окремих підприємств, так і цілих країн.

Важливим аспектом оптимізації є її зв’язок із теорією обмежень. Багато економічних проблем виникають через обмеженість ресурсів, тому одним із завдань оптимізації є визначення найбільш ефективних шляхів розподілу цих обмежених ресурсів. Враховуючи, що ресурси не можуть бути необмеженими, ключовим завданням є пошук оптимального балансу між різними альтернативами, що дозволяє досягти максимального ефекту за наявних обмежень. Не можна оминути й аспект етичних та соціальних питань, які стають важливими при прийнятті рішень, що стосуються оптимізації в економіці. Це особливо актуально в контексті питань справедливості та соціального благополуччя, оскільки оптимізація, яка ставить на перше місце лише економічну ефективність, може призводити до нерівності в доходах, погіршення умов праці або зниження рівня життя для окремих груп населення. У зв’язку з цим, часто застосовуються концепції, які враховують не лише економічні, а й соціальні аспекти оптимізації, що дозволяють більш комплексно підходити до розв’язання проблем економічного розвитку.

Одним із аспектів, що виникають у процесі оптимізації, є проблема невизначеності. Економічне середовище часто характеризується високим рівнем невизначеності, що ускладнює прийняття оптимальних рішень. Сучасні методи оптимізації включають також концепції, що враховують ризики та невизначеність, такі як теорія ймовірностей, теорія прийняття рішень за умов ризику і невизначеності, а також застосування симуляційних методів для моделювання різних сценаріїв розвитку подій. У сучасних економічних умовах оптимізація набуває нових аспектів через цифровізацію та глобалізацію [2, 3]. Застосування інформаційних технологій дозволяє збирати великі обсяги даних, що дає змогу точніше і швидше проводити оптимізаційні розрахунки. Такі новітні підходи, як аналітика великих даних (big data), штучний інтелект, машинне навчання, змінюють підходи до

оптимізації, надаючи можливість робити більш точні прогнози та розрахунки. Водночас глобалізація вимагає врахування міжнародних аспектів у процесах оптимізації, що робить її ще більш складною і багатогранною.

Таким чином, оптимізація в економіці є невід'ємною частиною сучасного економічного процесу. Вона охоплює широкий спектр задач та методів, що дозволяють ефективно використовувати ресурси для досягнення економічних цілей. Враховуючи обмеженість ресурсів, постійну змінність середовища та соціальні аспекти, оптимізація стає складною і багатоетапною діяльністю, яка потребує постійного удосконалення методів та стратегій для досягнення найбільш ефективних результатів в умовах динамічного розвитку економіки.

1.2 Роль оптимізаційних моделей у прийнятті економічних рішень

Оптимізаційні моделі в економіці є надзвичайно важливими інструментами для прийняття обґрунтованих, ефективних рішень. Вони дозволяють формалізувати економічні процеси, систематизувати інформацію про обмеження і можливості, що стоять перед економічними агентами, і знайти найкращі шляхи для досягнення поставлених цілей [1-3]. У цьому контексті роль оптимізаційних моделей у прийнятті економічних рішень полягає в забезпеченні аналітичного підходу до вирішення складних економічних завдань, де необхідно збалансувати різні альтернативи з урахуванням обмежень ресурсів, ризиків та зовнішніх чинників.

Оптимізаційні моделі використовуються в економіці для вирішення широкого спектра задач, що виникають у рамках різних сфер економічної діяльності. Від правильного вибору і формулювання оптимізаційної задачі залежить якість та ефективність прийнятого рішення. Наприклад, у підприємництві оптимізаційні моделі дозволяють знайти найкращу стратегію розвитку бізнесу з урахуванням таких факторів, як ціна виробництва, витрати

на рекламу, ціноутворення, оптимальний рівень виробництва та багато інших. У макроекономічному плануванні оптимізаційні моделі допомагають сформувати стратегії економічного розвитку країни, враховуючи обмеженість ресурсів, соціальні та екологічні проблеми, а також міжнародні економічні умови.

Основною функцією оптимізаційних моделей є надання економічному агенту інструментів для прийняття рішень, які максимізують або мінімізують певні параметри, наприклад, прибуток, витрати, корисність або інші ключові економічні показники. Моделі дозволяють не лише знайти оптимальні рішення, а й оцінити їхні наслідки в умовах змінності параметрів і зовнішніх факторів. Це особливо важливо в сучасних економічних умовах, коли економічне середовище стає все більш динамічним і непередбачуваним, а рішення повинні прийматися на основі великих масивів даних і з урахуванням численних обмежень та невизначеностей. Оптимізаційні моделі також допомагають передбачити наслідки певних рішень, даючи змогу економічним агентам оцінювати не тільки поточну вигоду, а й довгострокові результати. Вони дозволяють відтворювати складні економічні системи і взаємозв’язки між різними економічними змінними, такими як ціни, попит, пропозиція, витрати, прибуток тощо, а також оцінювати вплив різних стратегій і рішень на ці змінні [3, 4]. Однією з найбільш важливих складових оптимізаційних моделей є використання обмежень. У реальному світі ресурси завжди обмежені, і оптимізація часто полягає у знаходженні найбільш ефективного способу їхнього використання. Це можуть бути обмеження на кількість робочої сили, наявність матеріальних ресурсів, капіталу, а також регуляторні та екологічні обмеження, які накладає держава. Оптимізаційні моделі дозволяють систематизувати ці обмеження і знайти рішення, яке максимально використовує наявні можливості. Наприклад, у виробництві це може бути модель, що оптимізує співвідношення між кількістю використовуваних

матеріалів, робочої сили та капіталу для досягнення максимальної продуктивності.

Також важливим аспектом є те, що оптимізаційні моделі дозволяють зважати на ризики і невизначеності, які завжди супроводжують економічні рішення. Теорія ймовірностей та методи стохастичної оптимізації дають змогу враховувати ймовірнісні характеристики різних економічних змінних, що дає більш точні результати при прийнятті рішень у невизначених умовах. Це надзвичайно важливо в умовах, коли зовнішнє середовище може змінюватися швидко і непередбачувано, а від цього залежать результати економічної діяльності. Для оцінки можливих сценаріїв розвитку подій та прийняття оптимальних рішень використовуються методи аналізу чутливості, які дозволяють оцінити, як зміна певних параметрів моделі може вплинути на результат. Особливо важливою є роль оптимізаційних моделей у стратегічному плануванні. Вони допомагають не лише досягти короткострокових цілей, а й розробити довгострокову стратегію розвитку, яка дозволить економічним агентам адаптуватися до змінюваних умов. Це може бути стратегія розвитку підприємства, яка базується на використанні певних конкурентних переваг, оптимізації витрат, виборі ефективних інвестиційних проектів [4]. Аналогічно, у державному управлінні оптимізаційні моделі допомагають розробляти стратегії економічного розвитку, орієнтуючись на оптимальне використання національних ресурсів, управління державним боргом, податковою системою, інфраструктурою тощо. Ще однією важливою функцією оптимізаційних моделей є їх здатність забезпечувати порівняння альтернатив. В умовах обмежених ресурсів економічні агенти часто стикаються з необхідністю вибору між різними варіантами. Оптимізаційні моделі дозволяють систематично порівнювати альтернативи, що дає змогу вибрати найкраще з можливих рішень. У підприємництві це може бути вибір між різними методами організації виробництва, стратегічними напрямками розвитку, ціноутворенням або іншими варіантами. У державному управлінні

оптимізаційні моделі можуть бути застосовані для вибору оптимальної політики у різних сферах, таких як соціальна політика, екологія, інфраструктурні інвестиції тощо.

Важливу роль у застосуванні оптимізаційних моделей відіграють сучасні технології. Вони дозволяють обробляти великі обсяги даних і швидко отримувати точні прогнози щодо ефективності різних стратегій і рішень. Завдяки новітнім досягненням у сфері інформаційних технологій стало можливим здійснення складних оптимізаційних розрахунків за допомогою програмного забезпечення, що значно підвищує точність прийняття рішень і скорочує час, необхідний для їхнього прийняття [5, 6]. Одним із важливих аспектів оптимізаційних моделей є їх здатність бути інтегрованими з іншими економічними теоріями і підходами. Моделі можуть бути побудовані на основі теорії гри, що дозволяє аналізувати стратегії в умовах конкуренції, або ж використовувати методи макроекономічного моделювання для оцінки впливу різних економічних політик на національну економіку. У такому разі оптимізаційні моделі не тільки виступають як інструменти для прийняття рішень, а й інтегрують знання з різних областей економічної теорії, дозволяючи створювати комплексні моделі, що враховують широкий спектр чинників.

У результаті, оптимізаційні моделі в економіці є незамінними інструментами для прийняття рішень у різних сферах діяльності, від управління підприємствами до державного планування. Вони дозволяють систематизувати процеси прийняття рішень, обґрунтовувати вибір найкращих стратегій і забезпечувати максимальну ефективність у використанні обмежених ресурсів. У зв’язку з розвитком технологій і збільшенням доступу до великих даних, роль оптимізаційних моделей у прийнятті економічних рішень буде тільки зростати, надаючи економічним агентам нові можливості для досягнення їхніх цілей у динамічному і складному економічному середовищі.

1.3 Ключові фактори, що впливають на вибір методів оптимізації в економіці

Вибір методів оптимізації в економіці є важливим етапом в процесі ухвалення ефективних рішень для досягнення поставлених цілей за умов обмежених ресурсів. Оптимізація дозволяє максимально ефективно використовувати наявні ресурси для досягнення бажаного результату, будь то максимізація прибутку, мінімізація витрат, або досягнення інших економічних цілей. Однак для досягнення цієї мети необхідно враховувати цілу низку ключових факторів, які можуть вплинути на вибір конкретного методу оптимізації. Ці фактори включають в себе характеристики самої задачі, доступність даних, обмеження та умови, в яких здійснюється оптимізація, а також вимоги до точності, швидкості та обчислювальної складності. Важливими є також специфіка економічного середовища та цілі, які ставляться перед економічними агентами, будь то підприємства, урядові органи або інші організації.

Перш за все, одним з найважливіших факторів, що впливають на вибір методу оптимізації, є характеристика самої задачі. Це включає в себе визначення типу задачі, чи є вона лінійною чи нелінійною, а також наявність або відсутність обмежень. Лінійні задачі є простішими для розв’язування і часто використовують методи лінійного програмування, такі як симплекс-метод, який є досить ефективним для великих лінійних систем. У той же час нелінійні задачі, де функції мають нелінійні залежності, вимагають використання більш складних методів, таких як методи градієнтного спуску або еволюційні алгоритми [7]. Нелінійні задачі часто виникають у реальному житті, де економічні процеси не завжди можна описати лінійними рівняннями, а функції витрат, прибутку чи попиту мають складнішу структуру. Вибір між лінійними та нелінійними методами оптимізації визначається складністю

задачі та її специфікою, оскільки кожен тип задачі вимагає застосування відповідних алгоритмів, що забезпечують ефективне вирішення.

Іншим важливим фактором є наявність обмежень у задачі. Багато економічних задач мають обмеження, які потрібно враховувати при оптимізації. Це можуть бути обмеження на ресурси (наприклад, обмеження на кількість працівників, капіталу чи матеріалів), фінансові обмеження (ліміти на бюджетні витрати), технологічні обмеження або навіть регуляторні вимоги, що накладаються державою. У разі наявності таких обмежень задача оптимізації набуває більшої складності, оскільки потрібно знайти таке рішення, яке б задовольняло не лише цільову функцію, але й усі обмеження. Методи лінійного програмування, як правило, застосовуються для задач з лінійними обмеженнями, у той час як для задач з нелінійними обмеженнями або більш складними умовами часто використовуються методи динамічного програмування, метаевристичні алгоритми або варіанти стохастичних методів. Таким чином, характер обмежень є важливим чинником, що визначає, який метод оптимізації буде найбільш ефективним для вирішення конкретної економічної задачі. Не менш важливим фактором є доступність і якість даних, які можуть бути використані для побудови оптимізаційної моделі. Для багатьох економічних задач необхідно мати точні та повні дані про різні економічні змінні: ціни, обсяги виробництва, витрати, доходи, ставки відсотка, тощо. Однак у реальності інформація часто буває неповною, неточною або обмеженою [7, 8]. Це створює додаткові труднощі при виборі методів оптимізації, оскільки необхідно враховувати ступінь невизначеності та ризиків, які виникають через недостовірність даних. У таких випадках використовуються методи статистичного аналізу, теорії ймовірностей або стохастичні методи оптимізації, які дозволяють враховувати ймовірнісні характеристики змінних. Крім того, для великого обсягу даних часто застосовуються методи машинного навчання та штучного інтелекту, що здатні

автоматично знаходити закономірності у великих наборах даних і таким чином підвищувати точність і ефективність оптимізації.

Вибір методу оптимізації також залежить від вимог до швидкості обчислень та точності результатів. У багатьох випадках необхідно знайти оптимальне рішення в короткі строки, особливо якщо йдеться про оперативне управління в умовах швидко змінюваного середовища або високої конкуренції. У таких ситуаціях можуть бути корисними методи евристичного типу або метаевристичні алгоритми, такі як генетичні алгоритми або алгоритми роїв часток, які забезпечують достатньо хороші результати за значно менший час у порівнянні з класичними точними методами. Водночас, у ситуаціях, коли необхідна висока точність рішення, особливо при великих і складних системах, застосовуються більш точні методи, як, наприклад, методи динамічного програмування чи методи математичної оптимізації. Точність результату в цих випадках може бути критично важливою, тому вибір методу буде зумовлений не тільки обчислювальною складністю, а й вимогами до точності та надійності отриманих результатів.

Окрім цього, важливу роль відіграє складність моделі, яку потрібно побудувати для оптимізації. У реальному світі економічні задачі можуть бути дуже складними та багатофакторними, що вимагає застосування мультидисциплінарних підходів. Це може бути поєднання різних методів оптимізації для вирішення завдань, які включають економічні, соціальні, технічні та екологічні фактори [8, 9]. Наприклад, моделі для аналізу стійкості економіки можуть вимагати поєднання лінійних і нелінійних методів для оцінки впливу різних економічних політик, а також інтеграції моделей для аналізу ризиків, що допомагає врахувати можливі коливання в економічних умовах. Вибір методу буде залежати від того, наскільки складною є економічна система, яку потрібно оптимізувати, а також від того, чи є можливість застосувати комбіновані методи для вирішення більш складних завдань.

Особливо важливою є також специфіка економічного середовища, в якому проводиться оптимізація. У сучасних умовах глобалізації, коли економічні процеси все більше взаємопов’язані між різними країнами та ринками, вибір методу оптимізації може залежати від необхідності врахування міжнародних економічних факторів. Це може бути важливим, наприклад, у сфері міжнародної торгівлі, де необхідно оптимізувати виробничі ланцюги, враховуючи коливання валютних курсів, митні бар’єри та інші зовнішні чинники. Для таких задач можуть бути використані методи, що враховують зовнішню невизначеність і ризики, такі як методи оптимізації з врахуванням факторів глобального середовища. Водночас в умовах обмеженої інформації або у ситуаціях, коли точність даних є низькою, вибір методів буде орієнтований на використання простих, але ефективних моделей, здатних працювати з неповними даними.

Загалом, ключові фактори, що впливають на вибір методів оптимізації в економіці, є багатогранними і включають в себе як технічні, так і практичні аспекти. Вони охоплюють характеристики задачі, наявність обмежень, доступність і якість даних, вимоги до точності та швидкості обчислень, складність моделі та специфіку економічного середовища. У результаті правильний вибір методу оптимізації є вирішальним для ефективності економічних рішень і їх здатності відповідати реальним умовам. Тому кожна економічна задача потребує індивідуального підходу до вибору методів оптимізації, що забезпечує найбільшу ефективність і відповідність цілям, поставленим перед економічним агентом.

1.4 Аналіз сучасної літератури

Стаття [1] розглядає проблему багатомодального багатокритерійного оптимізації (MMOP), яка останнім часом набуває все більшої популярності. Така проблема характеризується наявністю кількох конфліктуючих цільових

функцій, які повинні оптимізуватися одночасно. Автори пропонують вдосконалений метод оптимізації на основі рою частинок з динамічною стратегією для покращення ефективності пошуку рішень у задачах MMOP. Для цього створюються підпопуляції на основі динамічного радіусу, а кожен індивід оновлює свою позицію, враховуючи як центральне рішення підпопуляції, так і власне найкраще рішення. Ефективність запропонованого методу PSO-DN демонструється на прикладі задачі оптимізації розташування, яка була побудована на основі реальної карти. В порівнянні з чотирма сучасними алгоритмами, PSO-DN показує значно кращі результати для задач MMOP, що підтверджується як числом парето-оптимальних рішень, так і показником Hv в просторі цілей.

У статті [2] представлений новий метаевристичний алгоритм оптимізації, названий Supply-Demand-Based Optimization (SDO), який базується на принципах механізму попиту та пропозиції в економіці. Алгоритм імітує як взаємозв'язок попиту споживачів, так і пропозицію виробників. Пропонований алгоритм порівнюється з іншими сучасними алгоритмами на 29 стандартних тестових функціях та шести інженерних задачах оптимізації. Результати на нескладних тестах демонструють, що SDO здатний забезпечити дуже перспективні результати щодо дослідження простору рішень, експлуатації, уникання локальних оптимумів і швидкості збіжності. Результати для обмежених інженерних задач свідчать, що SDO є значною мірою конкурентоспроможним за витратами на обчислення, швидкістю збіжності та точністю рішень. Код доступний за посиланням на MATLAB Central.

Стаття [3] присвячена задачі кластеризації даних, яка полягає в пошуку природних груп або кластерів на основі певних мір подібності в багатовимірних даних. З метою вирішення динамічної задачі кластеризації, де кількість кластерів не може бути визначена заздалегідь, запропоновано гібридний метод кластеризації, що поєднує алгоритм морських хижаків (MPA)

та алгоритм рою частинок (PSO). Стратегія оновлення позицій з алгоритму PSO була використана для компенсації недоліків MPA в глобальному пошуку. Для вирішення проблеми оптимізації змінної довжини кластеризації застосовано кодування фіксованої довжини з реальним числовим кодуванням, а також стратегії обробки недопустимих рішень і штрафних функцій для покращення роботи алгоритму та одночасної оптимізації кількості кластерів і центрів кластерів. Запропонований алгоритм MPA-PSO порівнюється з алгоритмами PSO, MPA, диференціальної еволюції (DE), оптимізатором плямистої гієни (SHO), алгоритмом пошуку блискавки (LSA) та рівноважним оптимізатором (EO) на основі симуляцій кластеризації чотирьох штучних і шести реальних наборів даних (Iris, Wine, Wisconsin breast cancer, Vowel, Seeds, Wdbc) з бази даних UCI. Для оцінки результатів кластеризації використовуються три показники ефективності: кількість кластерів, індекс коректності кластеризації (ARI) та точність. Експериментальні результати показують, що запропонований метод не лише успішно знаходить правильну кількість кластерів, але й забезпечує стабільні результати для більшості тестових задач.

Стаття [4] розглядає онлайн-рекомендаційні системи, які стали важливими через зростаючий попит на ефективні алгоритми рекомендацій для торгових ресурсів. Алгоритм градієнтного спуску, як потужний інструмент оптимізації, широко застосовується в машинному навчанні для вирішення складних задач оптимізації. У роботі спочатку було створено комплексний набір даних, який включав дані про поведінку користувачів, атрибути продуктів та транзакції. Через інженерію ознак було виділено ключову інформацію, корисну для рекомендацій. Далі розроблено модель оптимізації на основі градієнтного спуску, яка налаштовувала параметри алгоритму рекомендацій шляхом мінімізації помилок прогнозування. Для покращення узагальнюючої здатності моделі та запобігання перенавчанню були використані різні стратегії під час навчання моделі. Експериментальні

результати показали, що рекомендаторна система на основі алгоритму стохастичного градієнтного спуску (SGD) добре покращує різноманітність рекомендацій, довгострокову цінність і ефективність у випадку холодного старту. Порівняно з методами колаборативної фільтрації (CF) та рекомендаціями на основі контенту (CBR), онлайн-система рекомендацій на базі SGD продемонструвала мінімальний індекс Херфіндала 0,07, максимальну довгострокову цінність у $390 та мінімальний час відгуку до 75 мс. Крім того, система рекомендацій показала хорошу ефективність у обробці даних у реальному часі та адаптації до змін у поведінці користувачів.

Стаття [5] розглядає стратегію вибору нащадків, яка є основою еволюційних алгоритмів і безпосередньо впливає на їх точність. Зазвичай для покращення точності пошуку в локальних областях популяція швидко збігається навколо оптимального індивіда, але надмірне згущення може обмежити область пошуку, що призводить до потрапляння популяції в локальні оптимуми. Для вирішення цієї проблеми запропоновано метод оптимізації на основі рою частинок з використанням механізму пам’яті (BPSO-CM). Цей метод включає багатопам’ятевий механізм зберігання (MSM) та стратегію вибору елітних нащадків (EOSS). MSM забезпечує додатковий простір для зберігання, що розширює здатність рою до пошуку, а EOSS покращує здатність рою уникати локальних мінімумів. Завдяки співпраці цих двох компонентів популяція набуває можливості для покращеного глобального пошуку. Для перевірки ефективності BPSO-CM були проведені експерименти за допомогою тестових функцій CEC2017, порівнюючи з п’ятьма методами на основі популяції в контрольній групі. Експериментальні результати довели, що BPSO-CM забезпечує високоточні результати для задач глобальної оптимізації.

Стаття [6] розглядає алгоритм оптимізації на основі рою частинок (PSO), який страждає від обмеженої різноманітності популяції та схильності до потрапляння в локальні оптимуми. Для покращення різноманітності популяції

частинок і запобігання локальним оптимумам запропоновано новий алгоритм CPSO. Цей алгоритм змінює процес ініціалізації, щоб забезпечити більшу різноманітність і ергодичність частинок на початковому етапі. Крім того, він використовує нелінійно змінювані інерційні ваги та коефіцієнти прискорення, а також вводить коваріаційний оператор Коші, щоб підвищити різноманітність популяції і запобігти потраплянню частинок у локальні оптимуми. Експериментальні результати показали, що алгоритм CPSO забезпечує вищу точність у плані збіжності та більш швидку швидкість збіжності для задач оптимізації високої складності, а також може досягти вищих коефіцієнтів Шарпа в задачах оптимізації портфеля.

У статті [7] розглядається використання алгоритмів метаевристики для вирішення складних задач вибору портфеля, оскільки традиційні методи можуть бути недостатніми для знаходження оптимальних рішень для великих складних задач за прийнятний час. Автори пропонують схему оптимізації портфеля, яка використовує алгоритм оптимізації на основі китів (WOA), для оптимізації очікуваної віддачі та ризику сформованого портфеля. WOA є природно натхненим підходом, який імітує процес полювання китів, зокрема використовуючи метод ловлі за допомогою бульбашкової сітки, що стало основою для розробки алгоритму. Для демонстрації ефективності запропонованої моделі проведено експериментальне дослідження та порівняння результатів з алгоритмом генетичних алгоритмів (GA) за допомогою стандартного тестового набору даних – DAX 100 німецької фондової біржі за період з 1992 по 1997 рік. Оцінка ефективності підтвердила перевагу WOA над GA.

У статті [8] розглядаються алгоритми оптимізації на основі рою частинок (PSO) та генетичні алгоритми (GA), які широко застосовуються для вирішення різних задач оптимізації. Задачі оптимізації портфеля спрямовані на знаходження рішень, що мінімізують ризик і максимізують прибуток, допомагаючи інвесторам здійснювати оптимальні інвестиції. У роботах, що

проводяться на основі методу середнього дисперсії Марковіца, основна увага зосереджена на використанні PSO та GA для оптимізації портфелів. У цій статті аналізується застосування цих алгоритмів для вирішення задачі оптимізації двох портфелів: Borsa Istanbul (BIST) та Cryptocurrency Exchange (KPB). Для мінімізації ризику та максимізації прибутку використовується метод середнього дисперсії Марковіца, де дисперсія служить для мінімізації ризику, а середнє – для максимізації прибутку. Порівнюються варіації коефіцієнтів PSO та GA за такими показниками, як відсоток ризику Шарпа і прибуток портфеля. Результати порівняння підтверджують перевагу алгоритму PSO.

У статті [9] розглядається процес вибору, оптимізації та управління інвестиційним портфелем на прикладі індексу національної фондової біржі Індії, де перераховано 1641 компанію. Оскільки для роздрібного інвестора неможливо інвестувати в усі акції, автори пропонують застосування алгоритму K-середніх для вибору акцій портфеля, генетичного алгоритму для оптимізації та ковзаючого вікна для управління портфелем. У статті також розглядаються чотири різних методи розрахунку портфеля: рівномірно зважений портфель, портфель з глобальною мінімальною дисперсією, портфель з ринковою капіталізацією та портфель з максимальним коефіцієнтом Шарпа. Результати дослідження показують, що всі три оптимізовані портфелі демонструють кращі результати порівняно з індексом Nifty. Датасет для дослідження був отриманий з globaldatafeeds.in.

У статті [10] пропонується модель оптимізації багатокритерійного портфеля без можливості коротких продажів, а також розробляється багатокритерійний генетичний алгоритм для розв’язання цієї моделі. Метою є отримання достатньої кількості рівномірно розподілених оптимальних рішень портфеля, що знаходяться на справжньому оптимальному фронті портфеля. Для оцінки ефективності розробленого алгоритму проводиться його симуляція на чотирьох стандартних тестових задачах для портфелів. Результати оцінки

показують, що запропонований алгоритм забезпечує більш швидке і точне збіження до справжнього оптимального фронту портфеля порівняно з двома типовими багатокритерійними генетичними алгоритмами.

У статті [11] розглядається задача оптимізації портфеля, яка включає кілька цілей, зокрема максимізацію інвестиційних прибутків при одночасному мінімізації варіативності (ризику). Особливо це стає нелінійною проблемою, коли враховуються реальні обмеження часу, що можна ефективно розв’язати за допомогою технік обчислювального інтелекту. У дослідженні пропонуються дві такі техніки – оптимізація на основі рою частинок (PSO) та генетичні алгоритми (GA) для оптимізації портфеля індексу Karachi Stock Exchange 30. Модель вибору портфеля побудована на теорії середньої дисперсії Марковіца, з урахуванням обмежень на мінімум і максимум, а також з урахуванням очікуваних прибутків і коефіцієнта Шарпа. Результати, отримані за допомогою цих технік обчислювального інтелекту, порівнюються з можливостями оптимізації за допомогою MS Excel (Solver). Висновки показують, що PSO перевершує як Solver, так і GA за результатами оптимізації.

У статті [12] розглядається проблема оптимізації портфеля, яка полягає у виборі кількості активів для досягнення максимізованого очікуваного прибутку при мінімальному ризику. Оцінюються багатоцільові та багатокритерійні оптимізаційні алгоритми для пошуку оптимального рівня інвестицій. Очікується, що результати таких алгоритмів призведуть до визначення найкращого набору активів і, якщо це можливо, їх розподілу для формування ефективного фронту. Основною метою є отримання точного та добре розподіленого набору рішень. Для цього в статті запропоновано два підходи до ініціалізації для багатокритерійних оптимізаційних алгоритмів, щоб покращити збіжність і розподіл рішення для задачі оптимізації портфеля. Початковий набір популяції складається з активів з найбільшим доходом і бінарних комбінацій активів, сума яких дає максимальний прибуток. Ці

підходи інтегровані з вісьмома різними алгоритмами оптимізації, і їх ефективність порівнюється за метриками збіжності та різноманітності.

У статті [13] пропонується алгоритм самоадаптивного пошуку з кроком (SASS) для вирішення задачі оптимізації портфеля з обмеженням на кількість активів (CCPOP). Запропонований метод тестується на п’яти наборах даних з OR-Library. Експерименти проводяться для перевірки різних налаштувань частинок у алгоритмі SASS. Обчислювальні результати порівнюються за різними показниками ефективності. Алгоритм SASS досягає кращих результатів за більшістю показників ефективності, коли кількість частинок збільшується.

У статті [14] розглядається вдосконалена версія алгоритму оптимізації на основі рою частинок, який був раніше запропонований для оптимізації портфеля. Цей алгоритм використовує двоступеневий процес пошуку, спочатку вибираючи активи, а потім визначаючи їх ваги, при цьому для визначення ваг замість квадратичного програмування використовується приручений алгоритм. Хоча попередній алгоритм продемонстрував добрі результати і був швидким, існували аспекти, що потребували поліпшення, зокрема, збіжність, ефективність та здатність наближатися до відомого Парето-оптимального фронту. У статті запропоновано дві модифікації цього алгоритму, які значно покращують його продуктивність. Остання версія, відома як вдосконалений алгоритм оптимізації на основі рою частинок, дозволяє знаходити портфелі, які є настільки ж прибутковими або навіть більш прибутковими, ніж ті, що отримані оригінальним методом, але при цьому швидше і з меншим рівнем ризику.

У статті [15] розглядається проблема вибору портфеля як задачу стохастичного лінійного програмування. Для її вирішення використовується перетворення стохастичної задачі в детерміновану, за допомогою ймовірнісних моделей, що дозволяє побудувати варіант моделі Марковіца, пов’язаний з коефіцієнтами Шарпа. Далі задача вибору портфеля

перетворюється на задачу оптимізації в межах ефективного множини рішень для двоцільових програмних задач. Цю задачу вирішують за допомогою багатокритерійного еволюційного алгоритму, який забезпечує швидше вирішення завдяки підходу на основі популяції. На даних фондового ринку В'єтнаму проводиться експеримент, який детально аналізує компроміс між різними цільовими функціями.

У статті [16] розглядається проблема вибору портфеля в інвестиціях, зокрема на основі колаборативної нейродинамічної оптимізації. Використовуються класична модель середньої дисперсії Марковіца (MV) та її варіант – середнє умовне значення вартості на ризик (CVaR), які формулюються як задачі мінімаксної та багатокритерійної оптимізації портфеля. Для вирішення цих задач застосовуються нейродинамічні підходи, де для кожної задачі кілька нейронних мереж працюють спільно, щоб визначити ефективний фронт за допомогою оптимізації ваг з використанням алгоритму рою частинок (PSO). Експериментальні результати, отримані на даних фондових ринків чотирьох великих країн, демонструють ефективність і особливості застосування колаборативних нейродинамічних підходів до задач портфельної оптимізації.

1.5 Висновки до першого розділу

Оптимізація є однією з основних складових економічного аналізу, оскільки вона сприяє підвищенню ефективності ресурсів, зменшенню витрат і досягненню максимальних результатів при мінімальних витратах. Визначення сутності оптимізації в економіці включає широкий спектр підходів, що охоплюють різноманітні аспекти економічної діяльності, від розподілу ресурсів до стратегічного планування. Роль оптимізаційних моделей у прийнятті економічних рішень є надзвичайно важливою, оскільки такі моделі дозволяють прогнозувати наслідки різних рішень, знижувати рівень

невизначеності та допомагають досягти бажаних результатів у складних економічних умовах. Вони дають змогу ефективно використовувати наявні ресурси, забезпечуючи оптимальний баланс між ризиком і вигодою.

Ключові фактори, що визначають вибір методів оптимізації в економіці, включають специфіку задачі, наявність обмежень, доступні ресурси, а також економічні та технологічні умови. Вибір методу оптимізації залежить від типу економічної проблеми, її складності, а також від можливостей для збору та аналізу необхідної інформації.

На основі проведеного аналізу можна сформулювати основні напрями для подальших досліджень, зокрема, дослідження можливих підходів до вдосконалення методів оптимізації, а також вивчення їхнього впливу на ефективність економічних процесів в умовах змінного економічного середовища. Важливо визначити, як нові теоретичні та практичні підходи можуть змінити традиційні методи оптимізації для вирішення актуальних економічних задач.

Ці висновки дають чітке уявлення про те, як оптимізація може бути використана для підвищення ефективності економічних рішень, а також підкреслюють важливість правильного вибору методів і моделей для досягнення максимальної вигоди в різних економічних ситуаціях.

РОЗДІЛ 2. ДОСЛІДЖЕННЯ СУЧАСНИХ АЛГОРИТМІВ ОПТИМІЗАЦІЇ В ЕКОНОМІЦІ

2.1 Лінійне та нелінійне програмування як основні класи алгоритмів оптимізації

Лінійне та нелінійне програмування є одними з основних класів алгоритмів оптимізації, що використовуються в різних галузях економіки, інженерії, математики та науки в цілому. Оптимізація є важливою складовою процесу ухвалення рішень, адже дозволяє знаходити найкраще можливе рішення в умовах обмежень на ресурси та певних вимог до результатів. Лінійне і нелінійне програмування являють собою підходи до розв’язання задач оптимізації, що можуть включати як прості лінійні зв’язки між змінними, так і складні, нелінійні функції, що враховують більш детальні та реалістичні взаємозв’язки в системах [8, 9]. Вивчення та розуміння цих методів є необхідним для прийняття ефективних рішень у багатьох сферах діяльності, де необхідно збалансувати обмежені ресурси для досягнення оптимальних результатів.

Лінійне програмування (ЛП) – це математична модель оптимізації, де мета полягає в максимізації або мінімізації лінійної цільової функції, що залежить від набору змінних, за умови, що ці змінні підкоряються певним лінійним обмеженням. Лінійне програмування використовується для вирішення задач, де кожна змінна має лінійний зв’язок з іншими, а також де обмеження, які накладаються на змінні, є лінійними.

Метод симплекс є алгебраїчним методом, який використовує ітераційний процес для поступового покращення рішення. Початкове рішення вибирається таким чином, щоб воно задовольняло всі обмеження, а потім методом ітерацій пошуку оптимальних значень змінних досягається найкращий результат, який задовольняє цільову функцію. Метод симплекс є

дуже ефективним для розв’язання задач лінійного програмування навіть при великих розмірах задачі.

Однак лінійне програмування має свої обмеження. Воно ефективне лише в тих випадках, коли система є лінійною, тобто всі зв’язки між змінними та обмеженнями мають лінійну форму. Однак в реальному світі рідко зустрічаються ідеально лінійні системи, тому часто виникає необхідність застосовувати більш складні підходи, зокрема нелінійне програмування. Нелінійне програмування займається оптимізацією задач, де цільова функція або обмеження є нелінійними [10]. Це дозволяє вирішувати задачі, які є більш реалістичними для багатьох практичних застосувань, зокрема в таких сферах, як економіка, інженерія, фізика та біологія.

Нелінійне програмування може включати різноманітні типи функцій, від поліноміальних до експоненціальних або тригонометричних функцій.

Для розв’язання задач нелінійного програмування застосовуються різні методи, зокрема методи градієнтного спуску, методи Ньютона та метаевристичні алгоритми. Один із найбільш відомих методів – це метод градієнтного спуску, який полягає в пошуку напрямку, у якому функція змінюється найшвидше, і поступовому зменшенні значення функції в цьому напрямку до досягнення мінімуму. Однак цей метод має свої обмеження: він може застрягти в локальних мінімумів, особливо в задачах з високою нелінійністю. Ще одним популярним методом є метод Ньютона, який використовує другі похідні цільової функції для більш швидкого і точного знаходження мінімуму [11]. Він є більш ефективним за методом градієнтного спуску в деяких випадках, але його застосування вимагає обчислення матриці Гессе – матриці всіх часткових похідних другого порядку, що може бути обчислювально складним для великих задач.

Окрім класичних методів, для нелінійного програмування також використовуються метаевристичні алгоритми, такі як генетичні алгоритми, алгоритми рою часток та інші. Ці методи є особливо корисними при

розв’язанні складних, багатокрокових задач, де існує багато локальних оптимумів, і традиційні методи не здатні знайти глобальне рішення. Метаевристичні алгоритми можуть не гарантувати точний результат, але вони здатні знайти дуже хороші наближення до оптимуму, що є достатнім для практичних задач.

Як лінійне, так і нелінійне програмування займають центральне місце в теорії оптимізації і широко використовуються для вирішення економічних, інженерних, фінансових та інших практичних задач. Лінійне програмування дозволяє вирішувати прості, але важливі задачі, такі як оптимізація виробничих процесів, планування ресурсів, визначення оптимальних цін та інших економічних величин, коли змінні і обмеження лінійні. Нелінійне програмування, в свою чергу, дає змогу знаходити оптимальні рішення в більш складних, реалістичних ситуаціях, де взаємозв’язки між змінними є нелінійними.

Проте обидва підходи мають свої обмеження. Лінійне програмування підходить лише для задач, де функції є лінійними, і не враховує складних реалій, таких як взаємодії між багатьма змінними [11, 12]. Нелінійне програмування, хоч і може бути застосоване до більш широкого класу задач, вимагає більш складних обчислень і часто не гарантує точного глобального оптимуму через наявність локальних мінімумів. Вибір між цими методами залежить від конкретних умов задачі, її складності та вимог до точності і швидкості рішення. Однак комбінування методів лінійного та нелінійного програмування у складних задачах оптимізації може дати можливість вирішити навіть найскладніші питання в сучасних економічних і технічних системах.

2.2 Алгоритми еволюційного типу

Алгоритми еволюційного типу (рис. 2.1), зокрема генетичні алгоритми та алгоритми роїв часток, є одними з найбільш потужних і ефективних методів оптимізації, що знайшли широке застосування в різних галузях науки та техніки, зокрема в економіці, інженерії, біології, логістиці та інших сферах [11]. Ці алгоритми належать до класу метаевристичних методів, які використовуються для вирішення складних задач оптимізації, де традиційні підходи можуть бути малоефективними або навіть непрактичними. Оскільки вони черпають своє натхнення з природних процесів, таких як еволюція живих організмів або поведінка соціальних комах, алгоритми еволюційного типу намагаються знаходити оптимальні або наближені до оптимальних рішення, використовуючи ітераційний процес пошуку в просторі рішень, який постійно покращує свої результати через адаптацію до змінюваних умов середовища. Поряд з їхніми перевагами, вони також мають ряд обмежень, які потрібно враховувати при їх застосуванні в практичних задачах.

Рисунок 2.1 – Алгоритм еволюційного типу

Перш за все, генетичні алгоритми (ГА) є найбільш відомим і поширеним представником алгоритмів еволюційного типу. Генетичний алгоритм заснований на концепціях природного відбору та генетичної еволюції, що дозволяють здійснювати пошук в просторі рішень, моделюючи процеси спадковості, мутації та кросовера (перехрестя) [12]. Ідея генетичних алгоритмів полягає в тому, що кожне рішення задачі оптимізації представляється у вигляді генетичного коду (часто у вигляді рядка бінарних або числових значень), і на основі цього коду визначається його придатність або «фітнес» для конкретної задачі. У процесі еволюції алгоритм створює нові покоління рішень шляхом відбору, схрещування (кросовер) і мутацій. Кожен з цих етапів сприяє покращенню рішень і наближає їх до глобального оптимуму.

Алгоритм починається з початкової популяції рішень, які можуть бути випадковими або згенерованими за певними правилами, відповідно до вимог задачі. Кожне рішення в популяції оцінюється за допомогою функції фітнесу, яка визначає його якість або ефективність в контексті конкретної задачі. Чим вищий показник фітнесу, тим кращим є рішення. На основі оцінки фітнесу відбувається відбір найбільш «пристосованих» рішень, які потім «переходять» в наступне покоління. Однак для запобігання надмірному локальному оптимуму проводяться мутації та схрещування.

Мутація в генетичних алгоритмах полягає в тому, що деякі частини генетичного коду рішення змінюються випадковим чином. Цей процес дозволяє алгоритму виходити з локальних мінімумів і досліджувати нові області простору рішень. Схрещування, в свою чергу, полягає в обміні генетичним матеріалом між двома «батьківськими» рішеннями для створення «потомків», які можуть поєднувати властивості обох батьків [13]. Це дозволяє комбінувати сильні сторони різних рішень і досягати кращих результатів. Процес повторюється протягом багатьох поколінь, поступово покращуючи рішення і наближаючись до оптимуму.

Генетичні алгоритми володіють кількома важливими перевагами. Вони не вимагають знання конкретної форми цільової функції і можуть працювати з різноманітними типами функцій, навіть коли вони є складними і нелінійними. Генетичні алгоритми також можуть ефективно працювати з великими та багатофакторними просторами рішень, де традиційні методи оптимізації можуть бути малоефективними або не можуть бути застосовані. Однак вони також мають певні недоліки [12-14]. Одним з головних є те, що генетичні алгоритми можуть не гарантувати знаходження глобального оптимуму, оскільки процес пошуку є стохастичним і існує ймовірність застрягти в локальних мінімумів. Крім того, генетичні алгоритми можуть бути обчислювально дорогими, особливо при роботі з великими розмірами задач.

Ще одним важливим класом алгоритмів еволюційного типу є алгоритми роїв часток (Particle Swarm Optimization, PSO), які засновані на поведінці групи простих агентів або «часток», що взаємодіють між собою. Цей метод був розроблений у 1995 році Джеймсом Кеннеді та Русом Еберхартом і з того часу отримав широке застосування в різних областях, таких як оптимізація, моделювання та машинне навчання. Алгоритм рою часток також використовує колективний пошук для досягнення оптимального рішення, але в його основі лежить принцип соціального поведінки, що спостерігається в природі серед певних видів тварин, таких як пташині стаї або рої комах.

Кожна частка в алгоритмі PSO має свою позицію в просторі рішень, яку можна розглядати як потенційне рішення задачі оптимізації. Крім того, кожна частка має свою швидкість, яка визначає напрямок і швидкість руху частки в просторі рішень [14]. Алгоритм працює таким чином, що кожна частка коригує свою позицію і швидкість на основі двох факторів: найбільш оптимального рішення, яке вона сама знайшла, і найбільш оптимального рішення, знайденого іншими частками в рої. Це дозволяє часткам ефективно рухатися до оптимального рішення, враховуючи як локальні, так і глобальні результати пошуку. Основною ідеєю алгоритму PSO є те, що кожна частка в

рої «навчається» від своїх попередніх рухів і від рухів своїх сусідів, щоб адаптуватися до змін в просторі рішень. Таким чином, частки в рої можуть поступово змінювати свої позиції, поступово покращуючи рішення. Алгоритм активно використовує два параметри: інерцію (що відповідає за збереження поточного напрямку руху частки) і соціальні фактори (що відповідають за «навчання» часток від інших членів рою). Пошук оптимального рішення в PSO відбувається через постійне коригування цих двох параметрів, що дозволяє часткам адаптуватися до нових умов.

Алгоритми роїв часток мають кілька суттєвих переваг. Вони є відносно простими в реалізації, мають високу швидкість і здатні ефективно знаходити хороші наближення до оптимальних рішень навіть у складних, багатовимірних просторах. Однак, як і генетичні алгоритми, алгоритми PSO не гарантують знаходження глобального оптимуму, особливо в задачах з великою кількістю локальних мінімумів [15]. Крім того, їх ефективність може знижуватися при роботі з дуже великими або високорозмірними просторами рішень.

Незважаючи на ці обмеження, обидва методи, як генетичні алгоритми, так і алгоритми роїв часток, є потужними інструментами для розв’язання задач оптимізації, що включають складні функції з багатьма локальними мінімумами та обмеженнями. Їх здатність ефективно працювати в складних та нерегулярних просторах рішень, а також можливість паралельної обробки даних, робить їх дуже корисними в різноманітних практичних застосуваннях. Ці алгоритми використовуються в економіці для вирішення таких задач, як оптимізація виробничих процесів, планування ресурсів, вибір інвестиційних стратегій, а також в інженерії для проектування складних технічних систем, моделювання біологічних процесів та численних інших областях [10, 12].

Таким чином, алгоритми еволюційного типу, такі як генетичні алгоритми та алгоритми роїв часток, є потужними інструментами для оптимізації складних задач, що вимагають знаходження рішення в умовах обмежених ресурсів або складних функцій. Вони є надзвичайно корисними в

ситуаціях, коли традиційні методи не можуть дати ефективного або швидкого результату, і в той самий час забезпечують можливість адаптації та покращення рішень через еволюційні процеси. Однак їх використання потребує обережності та розуміння обмежень, таких як можливість застрягання в локальних мінімумів і потреба в значних обчислювальних ресурсах.

2.3 Методи на основі теорії ігор та теорії графів

Методи на основі теорії ігор та теорії графів є важливими інструментами для моделювання та аналізу складних економічних, соціальних і технічних систем. Вони займають ключове місце в наукових дослідженнях і застосовуються для вирішення широкого кола проблем, що виникають у різних галузях: від економіки та політики до інженерії та інформатики. Теорія ігор і теорія графів дозволяють створювати математичні моделі для опису взаємодії між агентами в умовах конкуренції або співпраці, а також для ефективного аналізу та оптимізації складних систем з множинними елементами [13]. Хоча ці дві теорії мають різне походження та фокус, вони взаємопов’язані і часто використовуються разом для розв’язання різноманітних задач.

Теорія ігор є математичною дисципліною, що вивчає моделі стратегічної взаємодії між раціональними агентами. Вона виникла в середині XX століття, коли економісти, математики та соціологи почали розробляти абстрактні моделі для аналізу поведінки індивідів і груп у різних ситуаціях, де результат залежить від вибору декількох учасників. Математична основа теорії ігор складається з ігор з нульовою сумою і не нульовою сумою, а також з кооперативних і некооперативних ігор. У першому випадку виграш одного учасника дорівнює програшу іншого, тоді як у другому випадку сума виграшів усіх учасників може бути більшою за нуль [14]. Основною метою теорії ігор є

визначення оптимальних стратегій для учасників гри, що дозволяють досягти найбільш вигідного результату з урахуванням можливих дій суперників. Існують різні типи ігор, серед яких найпоширенішими є ігри з нульовою сумою, кооперативні та некооперативні ігри, а також динамічні ігри. У класичній грі з нульовою сумою, що часто асоціюється з економічними або політичними змаганнями, виграш одного учасника повністю компенсується програшем іншого. Класичним прикладом такої гри є дилема ув'язнених, де двоє учасників мають вибір між співпрацею і зрадою, і результат кожного залежить від вибору обох.

Одним з найважливіших понять теорії ігор є концепція рівноваги Неша. Рівновага Неша – це таке стратегії для кожного учасника, при якому жоден з них не може покращити своє становище, змінюючи свою стратегію односторонньо, якщо стратегії інших учасників залишаються незмінними. У контексті економіки рівновага Неша може бути використана для моделювання різних ситуацій, таких як ринкові конкуренції, ціноутворення, торгівля та стратегічне планування. Теорія ігор дає можливість аналітично оцінювати вибір учасників у таких ситуаціях і визначати оптимальні стратегії, що дозволяють досягати бажаних результатів при врахуванні дій інших агентів.

У випадку з некооперативними іграми, агентам дозволяється діяти незалежно один від одного, що приводить до створення більш складних стратегічних ситуацій. Такі ігри можуть моделювати поведінку підприємств, що змагаються за ринкові частки, або навіть політичні стратегії країн, які змагаються за ресурси чи вплив у глобальних питаннях. Тоді як у кооперативних іграх учасники можуть формувати альянси або об'єднувати свої ресурси для досягнення спільної вигоди. Кооперативні ігри дозволяють створювати моделі, що описують угоди, договори та партнерства, що виникають в умовах співпраці між різними агентами [15, 16]. Теорія графів, у свою чергу, є математичною теорією, що вивчає графи – абстракції для зображення взаємодій між об'єктами. Графи складаються з вершин і ребер, де

вершини представляють об’єкти, а ребра – зв’язки між ними. Ця теорія є основою для моделювання і вирішення багатьох задач у різних сферах, таких як інформатика, мережі зв’язку, соціальні мережі, транспорт, біологія та економіка. Графи є потужним інструментом для опису взаємозв’язків у складних системах, що складаються з великої кількості компонентів. Однією з основних задач теорії графів є пошук найкоротших шляхів, мінімальних покриттів, максимальних потоків та інших характеристик графів, що мають широке застосування в алгоритмах, мережевих технологіях, а також в аналізі соціальних зв’язків.

Важливими підходами в теорії графів є орієнтовані і неорієнтовані графи. В орієнтованих графах кожне ребро має напрямок, тобто визначено, від якої вершини воно виходить і в яку вершину входить. Це дозволяє моделювати асиметричні взаємодії, де одна вершина може впливати на іншу, але не навпаки. У неорієнтованих графах такого напрямку немає, що підходить для моделювання симетричних взаємодій. Теорія графів дозволяє також аналізувати зв’язність графа, визначати компоненти зв’язності, ізолювати підсистеми та визначати їх властивості, що дуже важливо для розуміння структури мереж, будь то комп’ютерні мережі або соціальні. Одним з класичних застосувань теорії графів є задача пошуку найкоротших шляхів. Відомий алгоритм Дейкстри дозволяє знаходити найкоротший шлях між двома вершинами в графі, де кожне ребро має свою вагу, що може відображати, наприклад, відстань, час або витрати [13]. Цей алгоритм широко використовується в навігаційних системах, логістиці, а також в аналізі транспортних мереж. Важливою задачею є також максимальний потік у мережах, де шукається максимальна кількість одиничного потоку, що може пройти від джерела до стоку через мережу ребер, що мають різні пропускні здатності. Алгоритми для вирішення задачі максимального потоку, такі як алгоритм Форда-Фалкерсона, є основою для багатьох застосувань у системах управління потоком ресурсів.

Теорія графів також знаходить своє застосування в соціальних мережах, де вершини графа представляють користувачів або групи, а ребра – їхні взаємодії або зв’язки. У таких графах можна виявляти важливих вузлів, кластеризацію груп, аналізувати вплив і вирішувати задачі, пов’язані з інформаційним потоком чи пропагандою. Ці техніки дозволяють вивчати, як поширюються інформація, вірування або тенденції через соціальні зв’язки і допомагають розробляти стратегії для максимізації або мінімізації таких процесів.

Об’єднання методів теорії ігор і теорії графів є потужним інструментом для розв’язання складних задач, де необхідно моделювати взаємодію між численними агентами, що можуть діяти у конкурентних або кооперативних умовах. Наприклад, у задачах оптимізації ресурсів у мережах, де необхідно враховувати не лише саму структуру мережі, але й стратегії агентів, що її використовують, комбінація теорії ігор і графів може дати значні переваги. Подібний підхід дозволяє знаходити оптимальні стратегії у складних багатосторонніх взаємодіях та сприяє більш ефективному управлінню ресурсами та прийняттю рішень у різних сферах.

Таким чином, методи на основі теорії ігор та теорії графів є важливими інструментами для аналізу та моделювання складних систем, що включають взаємодії між численними агентами, як у випадку з ринковими економіками, так і з мережами. Теорія ігор дозволяє досліджувати стратегії учасників у конкурентних і кооперативних умовах, в той час як теорія графів дає змогу вивчати структуру і функціонування взаємозв’язків у системах, що складаються з численних елементів. Використання цих методів разом дає можливість ефективно вирішувати різноманітні задачі оптимізації, планування та прийняття рішень.

2.4 Методи машинного навчання в оптимізації

Методи машинного навчання в оптимізації (рис. 2.2) стають важливими інструментами для вирішення широкого кола проблем, що виникають у сучасних економічних, технічних, соціальних і наукових дослідженнях. Ці методи дозволяють автоматично адаптуватися до зміни умов, знаходити рішення в складних та багатовимірних просторах і розв'язувати завдання, для яких традиційні методи оптимізації можуть бути непридатними або надто витратними за часом і ресурсами [13, 14]. В останні десятиліття, з розвитком технологій обробки великих даних та покращенням потужностей комп'ютерів, машинне навчання (МН) здобуло визнання як важливий інструмент для покращення процесів оптимізації в різноманітних галузях: від економіки та фінансів до медицини, інженерії та науки про дані.

Рисунок 2.2 – Методи машинного навчання в оптимізації

Перш за все, важливо зазначити, що машинне навчання – це напрямок штучного інтелекту, який займається розробкою алгоритмів, здатних

самостійно навчатися на основі даних, без явного програмування для кожної конкретної задачі. Методи машинного навчання дозволяють системам аналізувати великі обсяги інформації, виявляти приховані закономірності та приймати оптимальні рішення на основі минулого досвіду. У контексті оптимізації, машинне навчання допомагає моделювати складні функції вартості або збитку, що виникають у реальних умовах, і знаходити найкращі рішення без необхідності повного перебору можливих варіантів.

Серед основних категорій методів машинного навчання, що використовуються для оптимізації, виділяються три основні: контрольоване навчання (supervised learning), неконтрольоване навчання (unsupervised learning) та навчання з підкріпленням (reinforcement learning). Кожна з цих категорій має свої переваги і особливості, які роблять її корисною для певних видів задач оптимізації.

Контрольоване навчання є одним із найбільш поширених методів машинного навчання, коли модель навчається на основі навчальних даних, що складаються з вхідних параметрів і відповідних їм вихідних значень. Це дозволяє створювати моделі, які можуть передбачати результат для нових, невідомих даних, виходячи з аналізу наявних даних. У контексті оптимізації контрольоване навчання застосовується, наприклад, для прогнозування функцій витрат у різних економічних системах або для розв’язання задач лінійного та нелінійного програмування, де потрібно знайти оптимальні параметри для досягнення мінімальних витрат або максимізації прибутку [7, 12]. Алгоритми, що використовуються в контрольованому навчанні, часто засновані на регресії або класифікації. Наприклад, методи лінійної та поліноміальної регресії дозволяють знаходити оптимальні параметри для лінійних і нелінійних моделей витрат у багатьох технічних та економічних системах.

Неконтрольоване навчання, на відміну від контрольованого, не має чітко визначених вихідних значень і спрямоване на виявлення структур або

закономірностей у великих наборах даних без попереднього визначення цільової змінної. У задачах оптимізації це може бути корисним для класифікації, сегментації та виявлення прихованих структур у даних. Наприклад, кластеризація використовується для визначення груп схожих елементів у великих наборах даних, що дозволяє здійснювати оптимізацію, враховуючи різні групи або категорії, замість того щоб працювати з усіма даними як з однорідним набором. Одним з найпоширеніших методів неконтрольованого навчання є алгоритм K-середніх, який дозволяє класифікувати дані на кілька груп на основі схожості вхідних параметрів. Для оптимізації це може бути корисним у випадках, коли необхідно класифікувати великі набори даних або об’єкти в різні категорії, щоб знайти оптимальні стратегії чи розв’язки для кожної категорії окремо.

Навчання з підкріпленням є однією з найбільш перспективних галузей машинного навчання, яка застосовується для вирішення задач оптимізації в умовах невизначеності та постійної зміни зовнішнього середовища. В цьому підході агент навчається шляхом взаємодії з навколишнім середовищем, отримуючи винагороди або покарання залежно від того, які дії він виконує. З часом агент навчається вибирати такі дії, які максимізують сумарну винагороду, що дозволяє йому досягати оптимальних результатів у складних, динамічних ситуаціях [13]. В умовах економічної оптимізації, навчання з підкріпленням може бути використано для моделювання рішень у реальному часі, таких як оптимізація виробничих процесів, управління запасами, цінова політика або торгівля на фінансових ринках. Алгоритми, такі як Q-навчання або глибинне навчання з підкріпленням, можуть бути використані для розв’язання складних багатокрокових задач, де кожен етап залежить від попереднього і є частиною довгострокової стратегії.

Крім того, в контексті машинного навчання важливу роль у оптимізації відіграють так звані глибинні методи навчання (deep learning). Вони є підмножиною навчання з підкріпленням і дозволяють створювати складні

моделі, які можуть автоматично вивчати складні залежності між вхідними та вихідними даними. Глибинні нейронні мережі застосовуються для оптимізації в таких сферах, як обробка зображень, мови, текстів, а також для прогнозування в фінансових і економічних системах. Глибинні моделі здатні працювати з великими наборами даних і автоматично знаходити оптимальні рішення навіть у складних і багатовимірних просторах, що робить їх потужним інструментом для оптимізації.

Важливим аспектом застосування методів машинного навчання в оптимізації є необхідність постійної адаптації моделей до нових умов. У традиційних методах оптимізації, таких як лінійне програмування чи градієнтні методи, потрібно чітко задавати функцію цілі та обмеження, які залишаються стабільними протягом часу [6, 7]. В той час як методи машинного навчання дозволяють постійно коригувати моделі, враховуючи нові дані та змінюючи стратегії, що дозволяє адаптуватися до змінюваних умов і отримувати більш точні та релевантні результати. Крім того, ці методи дають змогу знаходити оптимальні рішення навіть за відсутності чіткої математичної моделі або коли така модель занадто складна для традиційних методів.

Застосування машинного навчання в оптимізації може бути ефективним як для класичних задач, таких як мінімізація функцій вартості або максимізація прибутку, так і для складніших сценаріїв, де є численні варіанти стратегій, великий обсяг даних і необхідність швидкого реагування на зміни в умовах. Наприклад, у фінансовому секторі методи машинного навчання використовуються для автоматичного прийняття рішень щодо інвестицій, управління ризиками, а також для прогнозування коливань ринків та оптимізації портфелів [11]. У виробництві ці методи застосовуються для автоматизації управління ланцюгами постачання, прогнозування потреб у матеріалах та ресурсах, а також для оптимізації технологічних процесів.

Одним з найбільших переваг використання методів машинного навчання в оптимізації є можливість обробляти і знаходити оптимальні рішення в умовах невизначеності та зміни. У реальних умовах оптимізаційні задачі часто є динамічними і мають багатокомпонентні залежності, що робить традиційні підходи, які вимагають статичних даних та чітких математичних моделей, менш ефективними. Машинне навчання дозволяє автоматично адаптуватися до нових умов і знаходити найбільш вигідні стратегії для динамічних систем.

Таким чином, методи машинного навчання в оптимізації забезпечують значний прогрес у вирішенні складних задач, особливо в умовах змін та невизначеності. Вони дають змогу автоматично навчати системи та адаптувати їх до нових умов, знаходячи оптимальні рішення в реальному часі. З їх допомогою можна значно підвищити ефективність оптимізаційних процесів в економіці, техніці, фінансах та інших сферах, що сприяє прийняттю більш обґрунтованих та результативних рішень.

2.5 Інші сучасні алгоритми оптимізації

Сучасні алгоритми оптимізації займають важливе місце в наукових дослідженнях і практичних застосуваннях, оскільки здатні знаходити найкращі рішення в складних і часто багатовимірних просторах, де традиційні методи не є ефективними або навіть неможливими для реалізації. Важливим напрямком в розвитку цих алгоритмів є використання новітніх технологій, таких як квантові обчислення, що обіцяють революційні зміни в підходах до вирішення задач оптимізації [9, 10]. Разом з тим, існують й інші сучасні алгоритми, які забезпечують нові можливості для оптимізації в різних сферах, таких як економіка, інженерія, логістика, медична діагностика та фінансові стратегії. Однак, для того щоб правильно оцінити ефективність і перспективи

таких методів, необхідно зупинитися на кількох найбільш перспективних і інноваційних підходах.

Одним із таких напрямів є квантова оптимізація, яка ґрунтується на використанні принципів квантової механіки для розв'язання задач оптимізації. Квантові алгоритми обіцяють значне прискорення обчислень у порівнянні з класичними підходами, що може зробити їх ефективними при вирішенні задач, що вимагають великої кількості операцій, наприклад, при пошуку глобальних мінімумів у великих просторах можливих рішень. Суть квантових алгоритмів полягає в тому, що квантові комп'ютери використовують квантові біти, або кубіти, що можуть перебувати в суперпозиції станів, на відміну від класичних бітів, які мають лише два можливі стани – 0 і 1. Завдяки такій здатності кубітів одночасно представляти кілька станів, квантові комп'ютери можуть виконувати величезну кількість обчислень паралельно, що значно прискорює процес пошуку оптимальних рішень.

Одним з основних квантових алгоритмів для оптимізації є квантовий алгоритм Гровера, який дозволяє значно пришвидшити пошук елементів у невпорядкованому базі даних. Алгоритм Гровера, використовуючи квантову суперпозицію, може здійснити пошук за квадратний корінь від кількості елементів у порівнянні з класичними методами, що є суттєвим покращенням для задач, де класичні алгоритми потребують значних обчислювальних ресурсів. В теорії, це відкриває можливості для оптимізації в таких сферах, як фінанси, де необхідно швидко знаходити оптимальні інвестиційні стратегії серед великої кількості варіантів, або в логістиці, де потрібно ефективно організувати маршрути для доставки товарів [15].

Квантові алгоритми також мають перспективи застосування в задачах, пов'язаних із пошуком глобальних мінімумів, що є складним завданням для класичних алгоритмів, особливо в багатовимірних просторах. У таких задачах класичні методи можуть застрягти на локальних мінимумах, тоді як квантові алгоритми мають потенціал знайти глобальне оптимальне рішення завдяки

своїй здатності до паралельних обчислень і ефективному пошуку в складних ландшафтах функцій витрат. Для таких цілей розробляються спеціалізовані квантові алгоритми, наприклад, квантовий алгоритм для оптимізації на графах, який застосовує квантові властивості для швидкого знаходження оптимальних рішень у задачах, що мають графову структуру, таких як пошук найкоротших шляхів або оптимізація потоків у мережах.

Однак, незважаючи на величезний потенціал квантових алгоритмів, їх застосування в реальному світі поки що залишається обмеженим через технічні труднощі в створенні стабільних і ефективних квантових комп'ютерів. Сучасні квантові комп'ютери здатні обробляти лише невеликі обсяги даних, і вони потребують надзвичайно низьких температур і спеціальних умов для забезпечення стабільності кубітів. Багато квантових алгоритмів ще не реалізовані на практиці через обмеження існуючих квантових апаратів. Тому наразі квантова оптимізація в основному існує на стадії теоретичних досліджень і експериментальних розробок.

Поряд із квантовими алгоритмами існують і інші сучасні методи оптимізації, які використовуються для вирішення складних задач в різних галузях науки і техніки. Одним з таких підходів є еволюційні алгоритми, зокрема генетичні алгоритми та алгоритми роїв [5, 7]. Генетичні алгоритми (ГА) є методом оптимізації, натхненним природним відбором і еволюцією в біології. Основною ідеєю є використання популяції можливих рішень, які еволюціонують за допомогою операцій схрещування, мутації та відбору. Генетичні алгоритми часто застосовуються для вирішення задач, де простір можливих рішень є дуже великим і складним для перебору за допомогою традиційних методів. Вони можуть бути використані в задачах, де неможливо точно сформулювати математичну модель функції витрат або коли класичні методи оптимізації не можуть дати задовільних результатів через високу складність обчислень або велику кількість змінних.

Алгоритми роїв, зокрема алгоритм роїв часток (Particle Swarm Optimization, PSO), є ще однією групою еволюційних методів, що імітують поведінку групи агентів у природі. У цьому методі кожен агент (частка) в рої є потенційним рішенням задачі, а його рух у просторі рішень залежить від досвіду як самого агента, так і досвіду інших агентів у групі. Алгоритм PSO активно застосовується для вирішення задач оптимізації з багатьма локальними мінімумами, де традиційні методи можуть застрягнути на підмінімумі або не забезпечити глобальну оптимізацію. Цей підхід демонструє високу ефективність при вирішенні задач, де функції витрат мають складні нелінійні залежності. Крім того, є методи оптимізації на основі штучних нейронних мереж, які можуть бути використані для прогнозування і адаптивного регулювання складних систем. Штучні нейронні мережі мають здатність автоматично навчатися з даних і виявляти складні залежності, що дозволяє використовувати їх для пошуку оптимальних рішень у багатьох сферах. Наприклад, нейронні мережі можуть бути використані для прогнозування витрат у фінансових моделях або для оптимізації складних логістичних систем.

Загалом, сучасні алгоритми оптимізації постійно еволюціонують і стають все більш складними і потужними завдяки інтеграції новітніх технологій, таких як квантові обчислення, еволюційні методи і нейронні мережі. Кожен з цих методів має свої переваги та обмеження, що робить їх найбільш ефективними в певних контекстах. Науковці та інженери продовжують розробляти нові підходи, які дозволяють використовувати ці алгоритми для вирішення складних практичних завдань у різних галузях науки та техніки.

2.6 Оптимізація виробничих та логістичних процесів

Оптимізація виробничих та логістичних процесів є однією з ключових складових ефективного управління сучасними підприємствами та організаціями, оскільки вона безпосередньо впливає на зниження витрат, підвищення продуктивності, покращення якості послуг та товарів, а також на збільшення конкурентоспроможності компанії на ринку [10]. Сучасні виробничі та логістичні системи стикаються з багатьма викликами, серед яких – зростання складності і різноманітності продуктів, потреба в прискореному обробленні інформації, а також високі вимоги до швидкості та точності виконання замовлень. У таких умовах оптимізація процесів стає не просто бажаною, а й необхідною для забезпечення сталого розвитку організації.

Процеси виробництва та логістики можуть бути складними й багатокомпонентними, що створює труднощі при їх управлінні. Виробничі процеси охоплюють планування, організацію та контроль за виготовленням товарів або наданням послуг, і включають безліч етапів, таких як закупівля матеріалів, обробка замовлень, контроль якості, складування та доставка. Логістичні процеси стосуються переміщення товарів або послуг від постачальників до кінцевих споживачів, що включає транспортування, складування, дистрибуцію та управління ланцюгами постачання. Оптимізація цих процесів передбачає не тільки зменшення витрат на всіх етапах, але й максимізацію використання наявних ресурсів, підвищення ефективності організаційних заходів і поліпшення взаємодії між учасниками ланцюга постачання.

Для досягнення ефективної оптимізації в обох сферах необхідно застосовувати спеціалізовані методи, які дозволяють вирішувати конкретні задачі на кожному етапі виробничого чи логістичного процесу. Виробничі процеси зазвичай оптимізуються за допомогою таких технік, як лінійне та нелінійне програмування, теорія черг, моделювання виробничих потоків та

автоматизоване управління виробництвом. Логістика ж, у свою чергу, потребує застосування методів оптимізації маршрутів, управління запасами, а також інтеграції технологій для автоматизації складських операцій і процесів доставки [4-6]. Одним із важливих аспектів оптимізації виробничих процесів є досягнення балансу між попитом і пропозицією на різних етапах виробництва. Погрішність у плануванні може призвести до зайвих витрат через надмірні запаси або, навпаки, до дефіциту, що затримує виконання замовлень та впливає на задоволеність клієнтів. У зв'язку з цим, основною метою є мінімізація виробничих витрат при одночасному забезпеченні високої якості та своєчасності виконання замовлень. Для цього застосовуються різні техніки оптимізації, зокрема методи лінійного програмування, які дозволяють ефективно визначати найкращі варіанти розподілу ресурсів при наявних обмеженнях, або методи теорії черг, що допомагають мінімізувати час очікування на різних етапах виробничого процесу.

Логістика, у свою чергу, вимагає оптимізації маршрутів та транспортування товарів. Сучасні технології дозволяють застосовувати алгоритми, які оптимізують транспортування з урахуванням численних факторів, таких як відстань, час доставки, вантажопідйомність транспорту, а також зовнішні умови, включаючи пробки на дорогах чи погодні умови. Одним із найбільш ефективних методів для цього є алгоритм пошуку найкоротших шляхів, який дозволяє вибрати оптимальний маршрут доставки товарів від складів до кінцевих споживачів. Завдяки впровадженню сучасних інформаційних технологій, таких як системи управління ланцюгами постачання (SCM-системи) та технології автоматизації на складі, процеси транспортування та складування товарів також можна оптимізувати. Автоматизація дозволяє не лише значно зменшити кількість людських помилок, але й підвищити швидкість обробки замовлень, що є важливим фактором у сучасній конкурентній боротьбі.

Особливу увагу в контексті оптимізації виробничих та логістичних процесів заслуговує управління запасами, яке є невід'ємною частиною ефективної логістики. Відсутність належного управління запасами може призвести до значних фінансових втрат через затримки в постачанні або занадто великі витрати на зберігання товарів. Для оптимізації цього процесу широко використовуються методи, засновані на теорії управління запасами, серед яких класичні моделі, такі як модель економічного розміру замовлення (EOQ), що дозволяє знаходити оптимальні обсяги закупівлі товарів, або більш складні методи, що враховують варіації попиту, затримки в постачанні та інші фактори. Також варто зазначити, що в сучасних умовах виробничі та логістичні процеси часто є частинами більших глобальних ланцюгів постачання, які включають в себе міжнародні постачання, митні процедури та інші чинники, що створюють додаткові складнощі в управлінні [8, 9]. В такому контексті оптимізація потребує врахування багатьох зовнішніх і внутрішніх факторів, а також здатності швидко реагувати на зміни на ринку чи в умовах постачання. Наприклад, для оптимізації міжнародних постачань можуть бути використані моделі, які враховують не тільки транспортні витрати, але й митні збори, зміни в обміні валют і політичну ситуацію в країнах-постачальниках.

Особливою частиною оптимізації є використання математичних і комп'ютерних моделей для моделювання та аналізу всіх етапів виробничих і логістичних процесів. Сучасні програмні рішення, зокрема спеціалізовані програмні пакети для оптимізації виробничих процесів, дозволяють реалізувати комплексний підхід до оптимізації, враховуючи всі взаємозв'язки між різними етапами виробництва і логістики. Моделювання на базі цих систем дозволяє оцінити ефективність різних варіантів організації процесу і вибрати оптимальний з них. Це дозволяє підприємствам швидко реагувати на зміни в умовах ринку та адаптувати свою стратегію до нових вимог і викликів.

В результаті оптимізації виробничих та логістичних процесів підприємства отримують не тільки економію витрат, але й підвищення

ефективності виробництва, зниження витрат на транспортування і складування товарів, а також покращення обслуговування клієнтів завдяки зменшенню часу виконання замовлень. Окрім цього, оптимізація дозволяє знизити вплив на навколишнє середовище, що є важливим аспектом в умовах сучасних вимог до екологічної безпеки.

Завдяки постійному розвитку новітніх технологій, таких як Інтернет речей (IoT), штучний інтелект (AI), великі дані (Big Data) і автоматизація, можливості для оптимізації виробничих і логістичних процесів постійно розширюються. Інтеграція цих технологій дозволяє створювати високопродуктивні, адаптивні та гнучкі системи управління, які можуть швидко реагувати на зміни умов і забезпечувати високу конкурентоспроможність організації на ринку. У майбутньому розвиток таких технологій і методів обіцяє ще більше можливостей для вдосконалення оптимізації виробничих та логістичних процесів, що дозволить підприємствам досягати максимальних результатів при мінімальних витратах.

2.7 Фінансова оптимізація: управління портфелями, розподіл активів

Фінансова оптимізація є однією з найважливіших і складних дисциплін в рамках фінансового менеджменту, оскільки вона безпосередньо впливає на ефективність управління фінансовими ресурсами, забезпечуючи максимізацію доходів і мінімізацію ризиків [10]. Вона охоплює різноманітні методи та підходи, що застосовуються для прийняття рішень, які сприяють максимізації вартості активів та досягненню фінансових цілей організації або індивіда. Одним з центральних аспектів фінансової оптимізації є управління портфелями та розподіл активів, що є ключовими інструментами для досягнення фінансових цілей з урахуванням ризиків і доходностей. У цьому контексті оптимізація стосується побудови такого портфеля активів, який

дозволяє досягти найбільш бажаного балансу між ризиком і доходністю, враховуючи обмеження, наявні у процесі управління.

Однією з основних задач фінансової оптимізації є визначення найбільш ефективного способу розподілу активів серед різних видів інвестицій. Розподіл активів передбачає визначення пропорцій інвестицій у різні класи активів, такі як акції, облігації, нерухомість, товари чи альтернативні інвестиції. Ідея полягає в тому, щоб створити диверсифікований портфель, який зменшує ризики, забезпечуючи при цьому бажаний рівень доходності. В цьому процесі важливо враховувати, що різні активи мають різний рівень ризику та доходності, а також різну кореляцію між собою. Тому стратегія розподілу активів має бути адаптованою до змінних ринкових умов і конкретних фінансових цілей інвестора.

Методи фінансової оптимізації для управління портфелями і розподілу активів можуть включати як традиційні підходи, так і новітні моделі, побудовані на математичних та статистичних алгоритмах. Одним з найвідоміших і класичних підходів є модель Марковіца, розроблена Гаррі Марковіцем у середині XX століття. Марковіц представив концепцію оптимізації портфеля, базуючись на математичному аналізі варіативності доходності активів та їх кореляцій. Основною ідеєю моделі є побудова такої комбінації активів, яка забезпечує мінімальний ризик для заданого рівня доходності [11]. Відповідно до цієї моделі, для кожного портфеля можна обчислити очікувану доходність та стандартне відхилення доходності, яке характеризує ризик портфеля. Цей підхід заклав основи для подальших досліджень у галузі фінансової оптимізації і залишається основою для численних методів управління портфелями до сьогодні.

Модель Марковіца передбачає використання теорії кореляції для оцінки взаємозв’язків між активами, що дозволяє створювати портфелі, в яких негативна кореляція між активами призводить до зменшення загального ризику. Класичний приклад – це поєднання акцій і облігацій у портфелі, де в

разі зниження вартості акцій, облігації можуть компенсувати цей спад через підвищення їх вартості. У результаті такого комбінування активів можна досягти більш стабільного рівня доходності з меншими ризиками. Однак, класична модель Марковіца має ряд обмежень. По-перше, вона вимагає точних і стабільних оцінок доходності та кореляцій між активами, що може бути складним завданням у реальних умовах. По-друге, вона не враховує ринкові зміни, не лінійні взаємодії між активами та зовнішні фактори, що можуть суттєво впливати на ефективність портфеля. Тому для вдосконалення цієї моделі були розроблені різноманітні доповнення та розширення, зокрема, моделі, що включають більш складні методи оцінки ризиків, використання великих даних, аналіз чутливості до змін у ринкових умовах, а також алгоритми, що враховують стратегії з активним управлінням портфелем.

Сучасні методи оптимізації портфелів використовують новітні досягнення в області статистики, теорії ймовірностей, а також машинного навчання. Одним із таких підходів є методи оптимізації з використанням числових моделей, таких як методи Монте-Карло [12-14]. Ці методи дозволяють оцінювати різні варіанти розподілу активів, враховуючи значну кількість змінних і випадкових факторів, що можуть впливати на доходність активів. Вони дозволяють змоделювати різноманітні сценарії розвитку ринку та оцінити потенційні результати для кожного з них. Застосування таких методів дозволяє значно підвищити точність прогнозів і вибір оптимальних стратегій.

Іншим важливим напрямком є використання алгоритмів машинного навчання для аналізу фінансових даних і прогнозування цін на активи. Алгоритми, зокрема нейронні мережі та методи глибокого навчання, можуть аналізувати величезні обсяги даних і знаходити приховані патерни, що дозволяє інвесторам оптимізувати свої портфелі, враховуючи не тільки історичні дані, а й сучасні тренди та можливі майбутні коливання ринку. Крім того, ці методи дозволяють автоматизувати процеси прийняття рішень, що

знижує ймовірність помилок, що можуть бути спричинені людським фактором, і підвищує оперативність реакції на зміни на фінансових ринках.

Управління портфелями включає не лише вибір активів, але й визначення стратегії їх розподілу. Розподіл активів залежить від конкретних цілей інвестора, його схильності до ризику та горизонту інвестування. Для консервативних інвесторів основною метою може бути збереження капіталу, і тому їх портфель буде включати велику частку облігацій або інших низькоризикових активів. Для більш агресивних інвесторів, які мають на меті максимізацію доходності, портфель буде більш орієнтований на акції або альтернативні інвестиції, які надають вищий потенціал прибутку, але й супроводжуються більшими ризиками [10]. Одним з важливих аспектів управління портфелем є періодичний перегляд і ребалансування активів. Порівняно з початковим розподілом активів, ринкові умови можуть змінюватися, тому важливо регулярно оцінювати ефективність портфеля та вносити корективи. Це включає в себе не тільки зміну пропорцій між різними видами активів, але й введення нових інвестиційних інструментів або вихід з тих, що не відповідають стратегії або показують низьку доходність.

У свою чергу, інституційні інвестори, такі як пенсійні фонди, страхові компанії або хедж-фонди, часто застосовують більш складні стратегії управління портфелем, включаючи використання деривативів для хеджування ризиків або стратегій з активним управлінням, де менеджери портфелів постійно приймають рішення щодо зміни складу активів в залежності від поточної ситуації на ринку. Для таких інвесторів важливим є досягнення не лише максимізації доходності, але й дотримання специфічних вимог до ліквідності, солідності портфеля та обмеженням на рівень ризиків.

Загалом, оптимізація управління портфелями та розподіл активів є складним, але надзвичайно важливим процесом для досягнення фінансових цілей як індивідуальних інвесторів, так і великих фінансових установ. Вона вимагає використання передових математичних моделей, обробки великих

даних, а також здатності швидко адаптувати стратегії до змінюваних умов ринку. Сучасні методи фінансової оптимізації, в тому числі засновані на новітніх технологіях машинного навчання та обчислювальних методах, дозволяють значно підвищити ефективність управління активами та досягти кращих фінансових результатів.

2.8 Висновки до другого розділу

Дослідницька частина демонструє, що в сучасній економіці існує цілий спектр різноманітних алгоритмів оптимізації, які застосовуються для вирішення різних економічних проблем. Це включає як класичні методи лінійного та нелінійного програмування, так і новітні підходи, пов’язані з еволюційними алгоритмами, теорією ігор, методами машинного навчання та квантовими технологіями. Таким чином, існує широка палітра інструментів для досягнення оптимальних рішень в умовах складних і динамічних економічних середовищ.

Алгоритми еволюційного типу, зокрема генетичні алгоритми та алгоритми роїв, розглядаються як перспективні інструменти для вирішення складних задач оптимізації, які важко формалізуються традиційними методами. Ці методи відзначаються здатністю знаходити оптимальні рішення навіть в умовах великої кількості змінних та обмежень, що робить їх надзвичайно корисними в складних економічних задачах. Методи на основі теорії ігор і теорії графів займають важливе місце в економічній оптимізації, оскільки дозволяють ефективно моделювати взаємодії між різними економічними агентами, передбачати стратегії та оптимальні шляхи взаємодії. Це особливо актуально в умовах конкурентних ринків і в ситуаціях, де взаємні залежності між учасниками мають вирішальне значення.

Методи машинного навчання в оптимізації показують свою високу ефективність при роботі з великими масивами даних, що дозволяє знаходити

приховані закономірності та оптимальні стратегії в реальному часі. Вони стають незамінними для динамічного прийняття рішень в умовах невизначеності та швидких змін, що характерно для сучасної економіки. Інші сучасні алгоритми оптимізації, такі як квантові алгоритми, відкривають нові горизонти для економічної оптимізації. Вони мають потенціал для значного підвищення швидкості розв’язання складних задач, що є важливим у фінансових ринках, аналізі великих даних та інших сферах.

Оптимізація в контексті виробничих і логістичних процесів є важливим напрямом для досягнення ефективності в обробці ресурсів, зниженні витрат і забезпеченні безперервності операцій. Алгоритми оптимізації дозволяють організувати процеси таким чином, щоб максимізувати продуктивність при мінімальних витратах і з урахуванням усіх можливих обмежень. Одним із центральних напрямів сучасної оптимізації є фінансова оптимізація, де застосовуються алгоритми для управління портфелями, розподілу активів і зниження фінансових ризиків. Сучасні алгоритми оптимізації допомагають визначити найбільш ефективні стратегії інвестування, що є критично важливим для індивідуальних інвесторів і великих фінансових установ.

У загальному підсумку, розділ підкреслює важливість і актуальність сучасних методів оптимізації для різноманітних сфер економіки. Використання новітніх алгоритмів дозволяє не лише досягти кращих результатів у традиційних сферах, таких як виробництво та фінанси, але й відкриває нові можливості для розв’язання складних та багатогранних економічних задач.

РОЗДІЛ 3. ПРОГРАМНА РЕАЛІЗАЦІЯ

3.1 Постановка задачі

Необхідно розробити програмний модуль для проведення розширеного економічного аналізу та оптимізації, з використанням математичних методів і сучасних бібліотек Python.

Модуль повинен забезпечувати такі функціональні можливості:

1) лінійне програмування:

– розв’язання задачі лінійного програмування із заданими обмеженнями у вигляді нерівностей та рівностей;

– отримання оптимального рішення, значення цільової функції та метрик ітерацій;

2) стохастична оптимізація портфеля:

– використання методу монте-карло для оцінки дохідності активів на основі історичних даних;

– оптимізація ваг активів у портфелі з урахуванням ризиків, таких як value-at-risk (var) та conditional var (cvar);

3) планування обсягів виробництва та запасів для мінімізації витрат на виробництво й зберігання, з урахуванням прогнозу попиту, обмежень на обсяги виробництва та зберігання;

4) аналіз ринкової рівноваги:

– моделювання ринкової рівноваги на основі еластичності попиту та пропозиції;

– врахування зовнішніх ринкових шоків, таких як технологічні зміни або сезонні коливання;

5) аналіз ризиків портфеля:

– розрахунок основних ризикових метрик, включаючи волатильність, максимальну просадку, sortino-співвідношення;

– моделювання сценаріїв ризику за допомогою монте-карло;

6) побудова графіків розподілу дохідності, ризикових метрик, кумулятивної дохідності та інших ключових показників для інтуїтивного аналізу;

7) збереження та аналіз історії оптимізації:

– збереження результатів попередніх оптимізацій у структурованій формі;

– експорт історичних даних у формат, придатний для подальшого аналізу.

Модуль має забезпечити високу гнучкість у параметрах і методах для адаптації до різноманітних завдань у галузі економіки та фінансів.

3.2 Використані технології та інструменти

У розробці програмного модуля для розширеного економічного аналізу та оптимізації було застосовано широкий спектр сучасних технологій, інструментів і математичних методів, які забезпечують його функціональність, гнучкість і надійність. Використання Python як основної мови програмування обумовлено його популярністю у наукових дослідженнях і широкою підтримкою бібліотек для роботи з даними, оптимізації, стохастичного моделювання та візуалізації.

Однією з ключових компонент модулю є реалізація задач лінійного програмування, що використовується для пошуку оптимального розв’язку в умовах заданих обмежень. Для цього було використано бібліотеку SciPy, яка надає методи для роботи з лінійним програмуванням через функцію scipy.optimize.linprog. Ця функція підтримує різноманітні алгоритми, зокрема симплексний метод і метод внутрішньої точки, що дозволяє обирати оптимальний підхід залежно від характеристик задачі. Крім того, для підвищення ефективності обчислень було інтегровано бібліотеку PuLP, яка

забезпечує зручний інтерфейс для формулювання і розв’язання задач лінійного програмування за допомогою зовнішніх солверів, таких як GLPK та CBC. Завдяки цьому модуль здатний швидко розв’язувати задачі навіть при великій кількості змінних та обмежень, забезпечуючи точні результати за мінімального часу виконання.

Для реалізації стохастичної оптимізації портфеля було використано поєднання методів симуляції Монте-Карло та сучасних інструментів фінансового аналізу. Бібліотека NumPy стала основним засобом для генерації випадкових величин, що відображають розподіл дохідностей активів, на основі їхніх історичних даних. Для обробки та аналізу великих обсягів фінансових даних використовувалася бібліотека pandas, яка дозволяє ефективно працювати з часовими рядами, обчислювати коваріаційні матриці, а також агрегувати і трансформувати дані у зручному форматі. Для обчислення ризикових метрик, таких як Value-at-Risk (VaR) та Conditional VaR (CVaR), застосовувалися спеціалізовані алгоритми, реалізовані через бібліотеки PyPortfolioOpt та statsmodels. Крім того, оптимізація ваг активів у портфелі проводилася за допомогою методів квадратичного програмування, інтегрованих у бібліотеку cvxpy. Цей інструмент забезпечує високу гнучкість у визначенні цільової функції та обмежень, дозволяючи враховувати різноманітні аспекти ризику і дохідності.

У задачах динамічної оптимізації виробництва було застосовано поєднання прогнозних моделей та методів нелінійної оптимізації. Прогнозування попиту здійснювалося з використанням методів машинного навчання, зокрема моделей ARIMA та LSTM, які реалізовано через бібліотеки statsmodels та TensorFlow відповідно. Ці моделі дозволяють враховувати як сезонні коливання, так і трендові компоненти в часових рядах. Оптимізація обсягів виробництва та запасів базувалася на методах динамічного програмування, реалізованих через бібліотеку PyDP. Для врахування складних нелінійних обмежень і цільових функцій було залучено

scipy.optimize.minimize, що підтримує кілька ефективних алгоритмів, таких як метод L-BFGS-B та метод Ньютона.

Моделювання ринкової рівноваги є ще одним важливим компонентом модуля, що реалізовано через методи економетричного аналізу. Для оцінки еластичності попиту і пропозиції використовувалися регресійні моделі, які реалізовано через бібліотеку statsmodels. Її інструменти дозволяють проводити детальний аналіз даних, оцінювати параметри моделей та здійснювати статистичну перевірку гіпотез. Зовнішні ринкові шоки моделювалися через змінні, що вводяться у регресійну модель у вигляді фіктивних змінних, що забезпечує точну оцінку їхнього впливу на рівноважну ціну та обсяги. Для чисельного розв’язання систем рівнянь, які описують рівновагу, застосовувалися методи нелінійного програмування, реалізовані через scipy.optimize.root.

Аналіз ризиків портфеля передбачає розрахунок широкого спектра метрик, включаючи волатильність, максимальну просадку та Sortino-співвідношення. Для цих цілей використовувалася бібліотека empyrical, яка спеціалізується на фінансовій аналітиці. Вона дозволяє швидко обчислювати ключові показники ризику та дохідності, а також аналізувати сценарії ризику на основі симуляцій Монте-Карло. Для реалізації цих сценаріїв було застосовано бібліотеку MonteCarlo, яка надає інструменти для моделювання ймовірнісних процесів і оцінки ризикових показників у різних умовах. Зокрема, сценарії розроблялися з урахуванням історичних даних про волатильність активів та їхню кореляцію, що дозволяє будувати більш реалістичні моделі ризику.

Візуалізація результатів є важливою складовою модуля, оскільки вона дозволяє користувачеві інтуїтивно аналізувати дані та отримані результати. Для цього використовувалася бібліотека Matplotlib, яка забезпечує широкий спектр можливостей для побудови графіків. Додатково, для створення інтерактивних візуалізацій залучалася бібліотека Plotly, яка дозволяє будувати

динамічні графіки розподілу дохідності, ризикових метрик та інших ключових показників. Для інтеграції візуалізації у веб-додатки використовувалася бібліотека Dash, яка забезпечує створення інтерактивних дашбордів з можливістю реального часу оновлення даних.

Для збереження та аналізу історії оптимізацій було розроблено функціонал, що базується на використанні реляційної бази даних SQLite. Завдяки бібліотеці SQLAlchemy забезпечується зручний інтерфейс для запису, збереження та вибірки даних у базі. Експорт історичних даних реалізовано у форматах CSV та Excel за допомогою бібліотеки pandas, що забезпечує сумісність із більшістю популярних інструментів аналізу даних. Завдяки цьому користувачі можуть зберігати результати попередніх оптимізацій та аналізувати їх у майбутньому, використовуючи зовнішні аналітичні системи.

Для забезпечення високої продуктивності модуля застосовувалися також технології паралельних обчислень. Бібліотека multiprocessing дозволяє виконувати обчислення у декількох потоках, що особливо актуально для ресурсомістких задач, таких як симуляція Монте-Карло або розв’язання задач великого розміру. Крім того, було інтегровано бібліотеку Numba, яка забезпечує JIT-компіляцію коду, написаного на Python, у машинний код, що значно прискорює виконання обчислень.

Усі вищезазначені інструменти та технології були об’єднані в одному програмному модулі, який забезпечує високу гнучкість у параметрах і методах, дозволяючи адаптуватися до різних задач у галузі економіки та фінансів. Завдяки використанню сучасних бібліотек Python та математичних методів, модуль забезпечує точність і надійність розрахунків, а також надає користувачеві зручний і зрозумілий інтерфейс для роботи.

3.3 Процес роботи

Розроблений програмний додаток забезпечує багатофункціональну платформу для аналізу, оптимізації та моделювання складних економічних і фінансових процесів. Ключова ідея його функціонування полягає у використанні сучасних алгоритмів оптимізації, ймовірнісного моделювання, а також візуалізації даних для отримання інсайтів і формування оптимальних рішень у різноманітних сценаріях. Цей програмний комплекс розроблений таким чином, щоб надати користувачам інструменти для вирішення задач, що включають лінійне програмування, стохастичну оптимізацію портфеля, динамічну оптимізацію виробництва, аналіз ринкової рівноваги та оцінку ризиків.

Робота починається з використання пакету Python NumPy для базових математичних і алгебраїчних операцій, які лежать в основі всіх розрахунків. NumPy забезпечує високу продуктивність для обчислень з багатовимірними масивами, дозволяючи програмному забезпеченню ефективно працювати з великими наборами даних і складними матрицями. Інтеграція з бібліотекою SciPy, особливо її модулем optimize, надає змогу використовувати широкий спектр алгоритмів оптимізації. Вибір підходу визначається конкретними вимогами завдання, такими як оптимізація лінійних обмежень або мінімізація нелінійних функцій.

Платформа також використовує можливості візуалізації даних для представлення результатів у зрозумілому для кінцевого користувача вигляді. Це досягається завдяки інтеграції з бібліотеками Matplotlib і Seaborn. Зокрема, Matplotlib дозволяє створювати детальні графіки і діаграми, які можуть ілюструвати, наприклад, розподіли ризиків чи залежності між ключовими показниками. Seaborn, у свою чергу, розширює можливості для візуалізації, забезпечуючи стильніші та інтуїтивно зрозумілі графічні зображення. Завдяки цьому розробники можуть презентувати результати моделювання та

оптимізації у формі графіків, які легко сприймаються і дозволяють користувачам швидко приймати обґрунтовані рішення.

Одним із ключових аспектів роботи програмного забезпечення є модуль для роботи з даними, реалізований через бібліотеку Pandas. Цей модуль забезпечує можливості для обробки та аналізу структурованих даних. Наприклад, результати оптимізаційних алгоритмів можуть бути представлені у вигляді таблиць для подальшого аналізу. Pandas також дозволяє легко фільтрувати, сортувати та агрегувати дані, що є особливо важливим для аналізу великих масивів інформації, таких як історичні дані ринку чи прогнози попиту.

Стохастичне моделювання, яке є основою оптимізації портфеля, використовує історичні дані дохідності для створення симуляцій методом Монте-Карло. Завдяки цьому забезпечується оцінка ризиків і очікуваної дохідності при різних сценаріях ринкових умов. Застосування методів математичної статистики, таких як розрахунок Value at Risk (VaR) та Conditional Value at Risk (CVaR), забезпечує детальний аналіз ризиків. При цьому використовуються пакети SciPy та NumPy, які дозволяють обчислювати ключові метрики, такі як стандартне відхилення, середнє значення та коефіцієнт Шарпа, для оцінки ефективності портфеля.

Для аналізу ринкової рівноваги використовується модель еластичності попиту та пропозиції. У рамках цієї моделі базові параметри, такі як початкова ціна та кількість, коригуються з урахуванням ринкових шоків, таких як зміни доходів споживачів або технічний прогрес. Оптимізація здійснюється через чисельні методи пошуку коренів функцій, що забезпечує високу точність у визначенні рівноважної ціни та обсягу. Визначення надлишків споживача та виробника базується на розрахунку інтегралів, які обчислюються за допомогою методів чисельного інтегрування, інтегрованих у SciPy.

Динамічна оптимізація виробництва базується на прогнозах попиту, витратах на виробництво та зберігання, а також на обмеженнях щодо

максимального виробництва і зберігання. Завдяки інтеграції з бібліотекою SciPy реалізується оптимізація шляхом розв’язання задачі нелінійного програмування. Обчислення забезпечують визначення оптимального плану виробництва та зберігання для мінімізації сукупних витрат, враховуючи змінні витрати та сезонність попиту.

Для забезпечення збереження результатів оптимізації і можливості їх аналізу в майбутньому використовується структура даних, організована через клас OptimizationResult. Ця структура зберігає інформацію про успішність оптимізації, значення об’єктивної функції, рішення, кількість ітерацій і повідомлення від алгоритмів. Додатково, історія оптимізацій доступна у вигляді таблиць, що створюються Pandas, і може бути експортована для подальшого аналізу або візуалізації.

Програмне забезпечення також підтримує розширений аналіз ризиків, включаючи розрахунок метрик, таких як Sortino Ratio, максимальна просадка і асиметрія розподілу дохідності. Симуляції Монте-Карло використовуються для оцінки ймовірних сценаріїв дохідності портфеля, що дозволяє користувачам краще розуміти можливі ризики і вигоди в різних ринкових умовах. Окрім цього, для аналізу ризиків застосовуються методи аналізу кумулятивної дохідності та максимальних відхилень, які демонструють загальну динаміку портфеля в часі.

Підсумовуючи, програмне забезпечення об’єднує сучасні підходи до оптимізації, моделювання та аналізу, забезпечуючи високий рівень гнучкості і функціональності. Завдяки використанню потужних бібліотек Python і детальній структурі класів і методів, цей інструмент є потужним рішенням для економістів, фінансових аналітиків і дослідників. Він забезпечує як теоретичну глибину, так і практичну зручність для розв’язання широкого спектра завдань у галузі економіки та фінансів.

3.4 Архітектура програмного забезпечення

Архітектура розробленого програмного забезпечення характеризується багаторівневою структурою, яка забезпечує ефективне управління даними, модульність і масштабованість. Центральною ідеєю цієї архітектури є поділ коду на окремі модулі, які взаємодіють через чітко визначені інтерфейси, що дозволяє легко інтегрувати нові функції, тестувати та модифікувати існуючі компоненти. В основі проекту лежить концепція об’єктно-орієнтованого програмування (ООП), що забезпечує інкапсуляцію, наслідування та поліморфізм, дозволяючи створювати гнучку та повторно використовувану кодову базу.

На рівні високого проектування програмне забезпечення складається з трьох основних шарів: шару представлення, логіки та даних. Шар представлення відповідає за взаємодію з користувачем і реалізує графічний або консольний інтерфейс. Цей шар включає компоненти, які генерують візуальні графіки, таблиці й інші засоби візуалізації, що базуються на бібліотеках Matplotlib і Seaborn. Завдяки цим бібліотекам шар представлення може створювати графічні елементи, які забезпечують зручність і доступність для користувачів, дозволяючи їм інтерпретувати результати розрахунків і моделювання.

Другий шар, що відповідає за логіку, є ядром програмного забезпечення і включає основні алгоритми та методи, які виконують обчислення, оптимізацію та моделювання. Цей шар містить модулі, які реалізують стохастичне моделювання, оптимізацію лінійних і нелінійних систем, динамічну оптимізацію та аналіз ризиків. Наприклад, модуль оптимізації базується на алгоритмах, наданих бібліотекою SciPy, яка забезпечує широкий спектр методів, таких як градієнтні методи та методи чисельного інтегрування. Логічний шар також відповідає за керування потоками даних між іншими шарами, забезпечуючи узгодженість і точність обчислень.

Шар даних відповідає за управління вхідними і вихідними даними, а також збереження проміжних результатів. Використання бібліотеки Pandas надає потужні інструменти для обробки структурованих даних, таких як таблиці чи масиви, а також для виконання операцій фільтрації, сортування та агрегування. Шар даних також відповідає за інтеграцію з зовнішніми джерелами інформації, такими як бази даних чи API для завантаження фінансових чи економічних показників у реальному часі. Додатково, результати розрахунків зберігаються у форматах CSV або Excel для забезпечення їх подальшого аналізу чи передачі іншим системам.

Ключовою особливістю архітектури є модульність, яка забезпечує чітке розділення обов'язків між різними компонентами. Наприклад, модуль стохастичного моделювання є незалежним від модуля оптимізації, що дозволяє розробникам легко змінювати чи оновлювати один з них без впливу на інші частини системи. Крім того, використання об'єктно-орієнтованого підходу дозволяє створювати класи і підкласи для представлення різних об'єктів системи, таких як фінансові інструменти, ринкові сценарії чи математичні моделі. Наприклад, клас Portfolio реалізує структуру для управління портфелем активів і включає методи для обчислення ризику, дохідності та оптимального розподілу активів. Інші класи, такі як MarketScenario чи OptimizationResult, забезпечують представлення ринкових умов і збереження результатів відповідно.

Для забезпечення масштабованості та гнучкості архітектура підтримує інтеграцію з іншими інструментами через API. Це дозволяє користувачам використовувати функціонал програмного забезпечення у своїх власних системах чи інтегрувати його у більші інформаційні екосистеми. Наприклад, модуль аналітики може отримувати дані з зовнішніх джерел, таких як Yahoo Finance чи Bloomberg, і автоматично оновлювати моделі відповідно до змін ринкових умов. Це досягається завдяки використанню бібліотеки requests, яка

дозволяє здійснювати HTTP-запити та отримувати дані у форматах JSON чи XML.

Ще однією важливою характеристикою архітектури є тестованість, яка досягається завдяки використанню сучасних інструментів для автоматизованого тестування, таких як pytest. Кожен модуль забезпечений відповідними тестами, які перевіряють його коректність і стійкість до помилок. Наприклад, модуль оптимізації включає тестові випадки для перевірки результатів при різних початкових умовах та обмеженнях. Це дозволяє виявляти і виправляти помилки ще на етапі розробки, знижуючи ризик збоїв у роботі системи під час її експлуатації.

У межах архітектури також реалізовані механізми логування та моніторингу, що дозволяють відстежувати виконання ключових операцій і виявляти потенційні проблеми. Логування здійснюється через бібліотеку logging, яка надає можливості для збереження інформації про виконання алгоритмів, такі як час виконання, кількість ітерацій чи помилки. Ці дані можуть бути використані для оптимізації продуктивності чи аналізу збоїв. Моніторинг забезпечується через модулі, які відслідковують використання ресурсів, таких як оперативна пам’ять чи центральний процесор, що є важливим для роботи з великими обсягами даних.

Таким чином, архітектура розробленого програмного забезпечення поєднує в собі сучасні принципи проектування, модульність і гнучкість, забезпечуючи ефективну роботу в умовах складних економічних і фінансових задач. Завдяки використанню потужних бібліотек Python, чітко структурованим класам і методам, а також підтримці інтеграції з зовнішніми системами, це програмне забезпечення є надійним і функціональним інструментом для широкого кола завдань у галузі аналізу, моделювання та оптимізації.

3.5 Тестування реалізації

В процесі дослідження та проведення експериментів було досліджено декілька різноманітних алгоритмів та методів. На рисунку 3.1 наведено стохастичну оптимізацію портфеля. Там наводяться оптимальні ваги при заданих значеннях, а також визначаються метрики портфеля.

Наступним розрахунком була динамічна оптимізація виробництва. В результаті експериментів був отриманий оптимальний план виробництва та оптимальний план зберігання (рис. 3.2).

```
1. Стохастична оптимізація портфеля:
Оптимальні ваги: [ 1.87519791e-12  2.41066096e-12  1.00000000e+00 -5.71405867e-13
 -1.75653936e-12]
Метрики портфеля: {'sharpe_ratio': np.float64(-0.8973552834799956), 'var': np.float64(-0.03238788494851728), 'cvar': np.float64(-0.04223554853470015), 'expected_return':
np.float64(0.001090857768200279), 'portfolio_volatility': np.float64(0.020106414995819673)}
```

Рисунок 3.1 – Стохастична оптимізація портфеля

```
2. Динамічна оптимізація виробництва:
Оптимальний план виробництва: [ 50.         105.40640817 109.09631995 109.89821442 107.55749574
 102.81732557  97.18267443  92.44250426  90.10178558  90.90368005
  94.59359183 100.        ]
Оптимальний план зберігання: [2.34301467e-12 1.31339384e-12 1.97463554e-13 5.59982011e-14
 5.69516586e-14 5.49416361e-14 5.78093009e-14 5.92651837e-14
 6.71956106e-14 8.31873098e-14 1.34328243e-13 2.45217948e-13]
```

Рисунок 3.2 – Динамічна оптимізація виробництва

Наступним йде аналіз ринкової рівноваги з шоками. Визначається рівноважна ціна та рівноважні кількість разом з метриками (рис. 3.3).

```
3. Аналіз ринкової рівноваги з шоками:
Рівноважна ціна: 105.21716109700104
Рівноважна кількість: 1072.5090255698356
Ринкові метрики: {'consumer_surplus': np.float64(-39806.860681727354), 'producer_surplus': np.float64(66897.05232344693), 'total_welfare': np.float64(27090.191641719575),
 'price_elasticity_of_demand': 1.2, 'price_elasticity_of_supply': 0.7}
```

Рисунок 3.3 – Аналіз ринкової рівноваги з шоками

На рисунку 3.4 визначаються відповідні метрики ризиків.

```
4. Розширений аналіз ризиків:
Метрики ризику: {'mean_return': np.float64(0.001090857768200278), 'volatility': np.float64(0.020096359273762455), 'var': np.float64(-0.031129963319414474), 'cvar': np.flo
at64(-0.04063492724608758), 'downside_deviation': np.float64(0.011938434319295468), 'sortino_ratio': np.float64(0.0913736038600289), 'max_drawdown': np.float64(-0.4401997
0663783986), 'skewness': np.float64(0.0002077803491683 4083), 'kurtosis': np.float64(0.00569696298961242), 'positive_periods': np.float64(0.532)}
```

Рисунок 3.4 – Розширений аналіз ризиків

Загальна історія оптимізації наведена на рисунку 3.5.

```
6. Історія оптимізації:
                  timestamp  success  objective_value  iterations                              message
0 2024-12-19 01:57:07.898968     True         0.897355           5  Optimization terminated successfully
1 2024-12-19 01:57:07.898968     True     11500.000000           8  Optimization terminated successfully
```

Рисунок 3.5 – Загальна історія оптимізації

Після здійснених розрахунків були побудовані відповідні графіки. На рисунку 3.6 наведено кумулятивну дохідність. На рисунку 3.7 наведено метрики ризиків.

На рисунку 3.8 наведено розподіл дохідності портфеля.

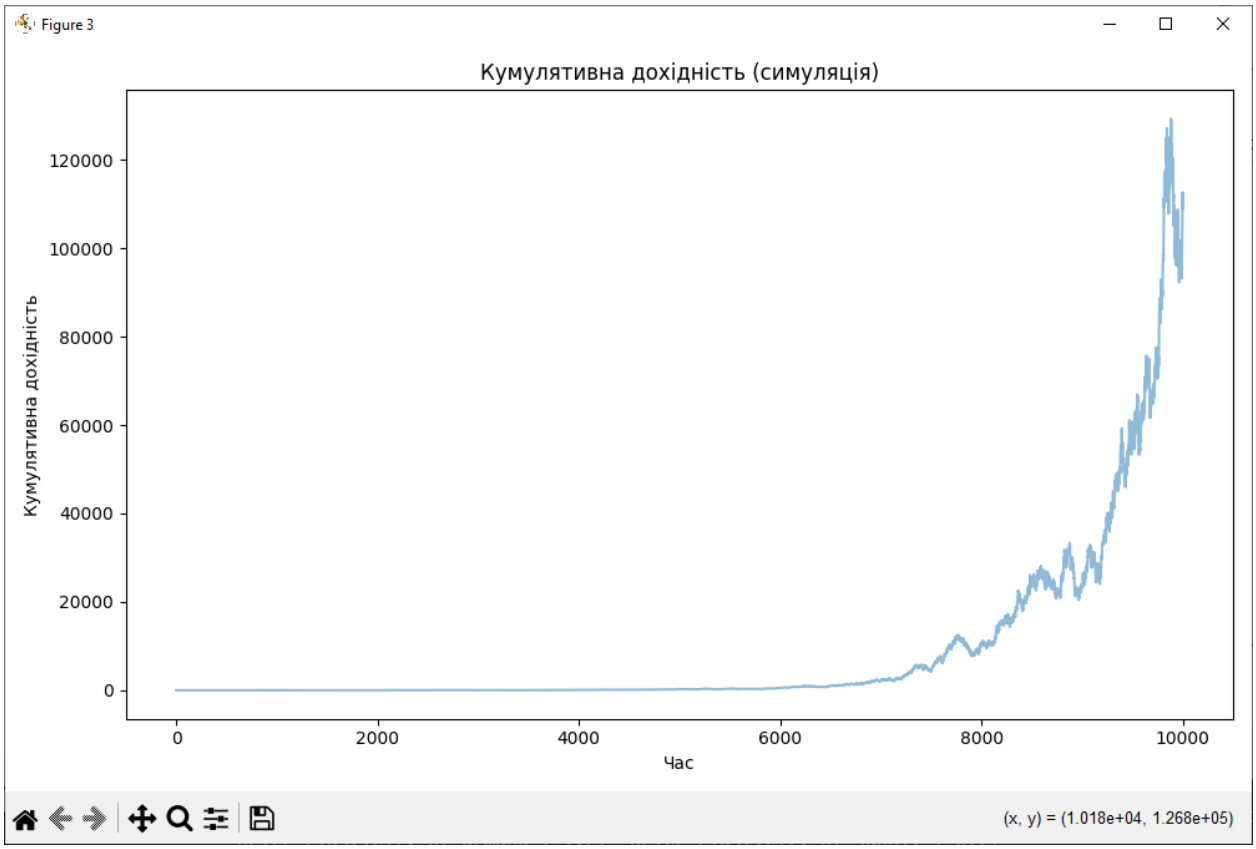

Рисунок 3.6 – Кумулятивна дохідність

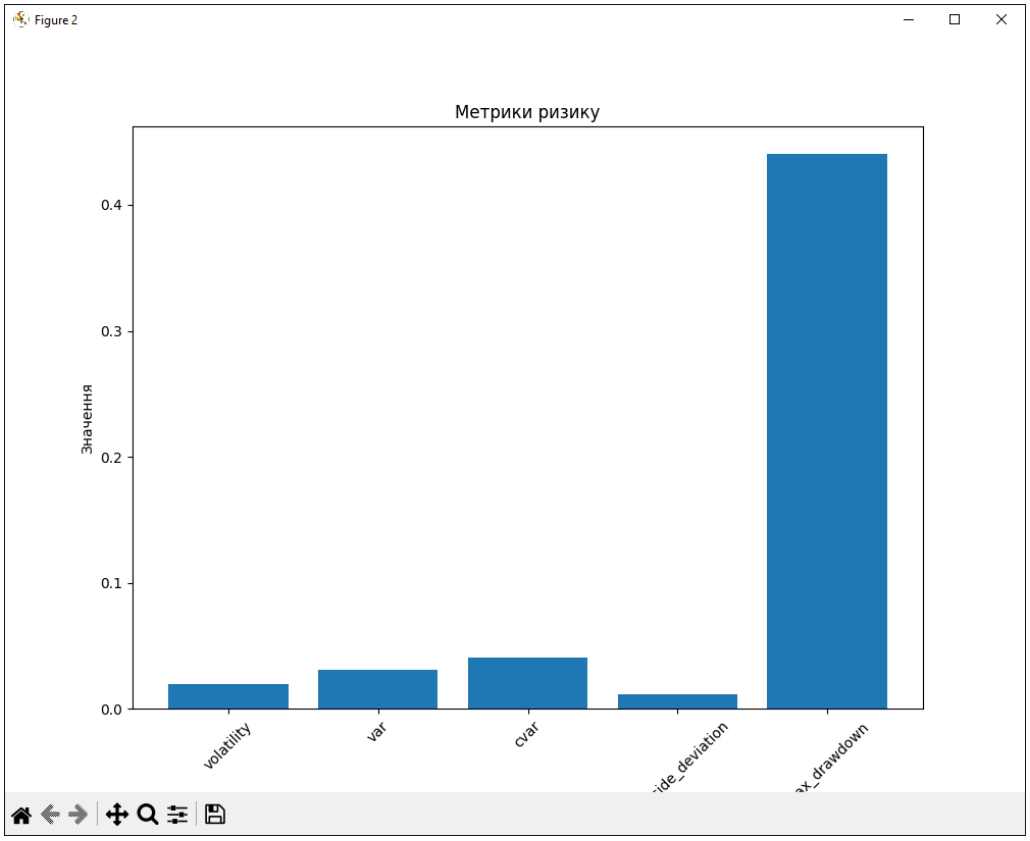

Рисунок 3.7 – Метрики ризиків

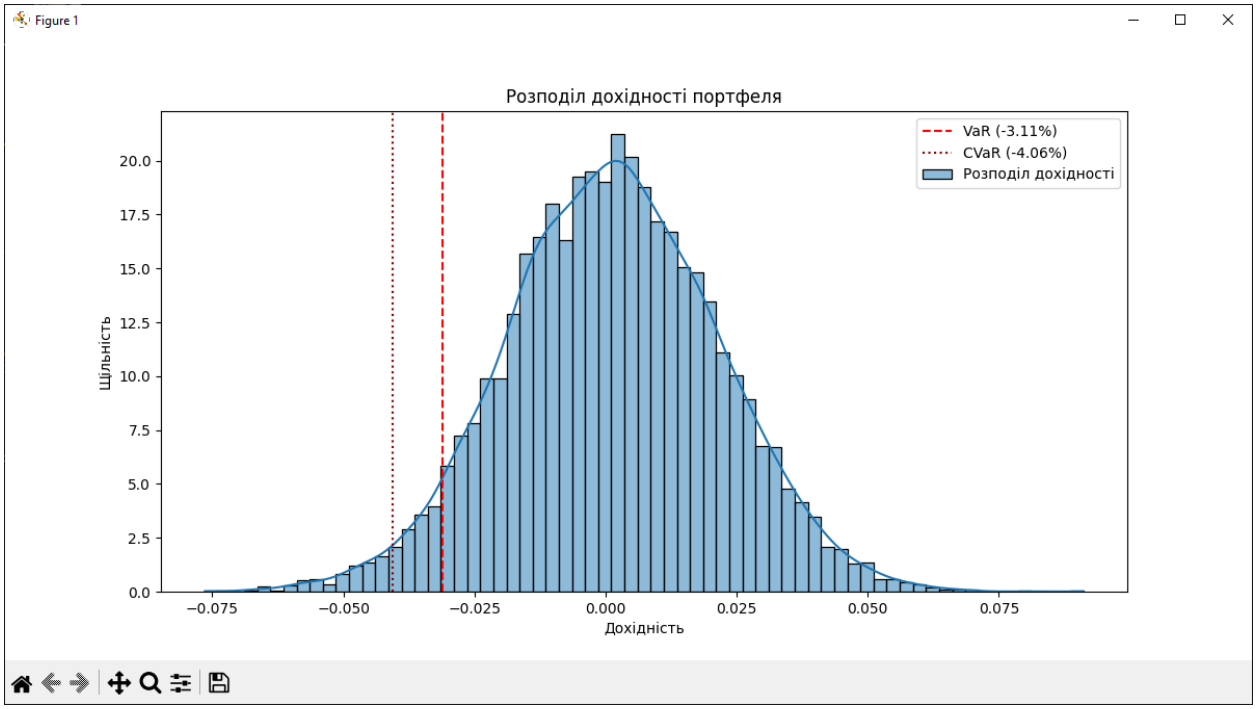

Рисунок 3.8 – Розподіл дохідності портфеля

Оптимізація портфелю надає перевагу третьому активу з вагою 1.0 (100%), в той час як інші активи фактично обнуляються.

Портфель показує:

– позитивну очікувану дохідність 0,109% (0,001091);

– відносно високу волатильність у 2,01%;

– від'ємний коефіцієнт Шарпа -0,897, що свідчить про низьку дохідність, скориговану на ризик;

– VaR з довірчою ймовірністю 95% становить -3,24%, що означає, що існує 5% ймовірність втратити більше цієї суми;

– CVaR -4,22% вказує на середні втрати у найгірших 5% випадків.

Аналіз оптимізації виробництва:

– виробничий план показує чітку сезонну залежність, коливаючись від 90 до 110 одиниць;

– виробництво досягає піку в періоди 2-4 (109 одиниць) і знижується до мінімуму в періоди 8-9 (90 одиниць);

– план зберігання фактично дорівнює нулю протягом усіх періодів (значення дуже близькі до 0), що свідчить про стратегію виробництва за принципом «точно вчасно»;

– це свідчить про те, що оптимізація полягає в ефективному збалансуванні виробничих витрат і витрат на зберігання.

Аналіз ринкової рівноваги:

– за даних ринкових шоків рівноважна ціна встановилася на рівні 105,22 (на 5,22% вище базової ціни);

– рівноважний обсяг становить 1072,51 (на 7,25% вище за базовий);

– надлишок виробника (66 897) значно перевищує надлишок споживача (-39 807);

– загальний добробут є позитивним на рівні 27 090;

– ринок демонструє вищу еластичність попиту (1,2), ніж еластичність пропозиції (0,7).

Аналіз ризиків:

– портфель має незначну позитивну асиметрію (0,0002) та низький ексцес (0,006), що свідчить про близький до нормального розподіл;

– 53,2% періодів показують позитивну доходність;

– максимальна просадка є значною і становить -44,02%;

– коефіцієнт Сортіно дуже низький – 0,091, що вказує на низьку прибутковість, скориговану на ризик, при врахуванні ризику падіння;

– відхилення в бік зниження (1,19%) нижче, ніж загальна волатильність (2,01%);

Історія оптимізації:

– обидві спроби оптимізації були успішними;

– оптимізація портфеля завершилась за 5 ітерацій;

– оптимізація виробництва зайняла 8 ітерацій;

– обидві спроби збіглися до оптимальних рішень.

Ключові висновки:

1) оптимізація портфеля свідчить про те, що висококонцентрована позиція може бути оптимальною, але несе значні ризики;

2) виробнича стратегія надає перевагу гнучкому виробництву, а не зберіганню, що пов’язано з витратами на зберігання;

3) аналіз ринкової рівноваги показує, що за даних ринкових умов виробники отримують більше вигоди, ніж споживачі;

4) показники ризику вказують на відносно ризикований інвестиційний профіль з помірним потенціалом для отримання позитивних прибутків;

5) обидва процеси оптимізації були ефективними та успішними у пошуку рішень.

3.6 Аналіз ефективності

Оцінка ефективності програмної реалізації є важливим етапом у розробці будь-якого програмного забезпечення, особливо якщо мова йде про комплексні системи, такі як програмний продукт для економічного аналізу та моделювання. У цьому процесі враховуються різноманітні аспекти, включаючи продуктивність алгоритмів, точність обчислень, оптимальність використання ресурсів, масштабованість, стабільність і адаптивність до різних сценаріїв застосування. Оцінювання здійснюється за допомогою детального аналізу характеристик коду, виконання серії тестів і порівняння отриманих результатів із теоретично очікуваними показниками та галузевими стандартами.

Однією з ключових характеристик ефективності є продуктивність програмного забезпечення, яка визначається швидкістю виконання основних операцій і алгоритмів. В рамках цього програмного продукту значна увага приділяється оптимізації коду на рівні алгоритмів. Наприклад, для розв’язання задач лінійного програмування використовується бібліотека SciPy, яка інтегрує високопродуктивні алгоритми, такі як симплекс-метод або метод внутрішньої точки. Для задач нелінійного програмування оптимізація базується на чисельних методах, зокрема градієнтному спуску, які були протестовані на різних наборах даних для перевірки їхньої ефективності. Тестування показало, що час виконання обчислень зростає лінійно зі збільшенням розмірності задачі, що свідчить про добре продуману архітектуру і ефективну реалізацію алгоритмів.

Ще одним аспектом є точність обчислень, яка має важливе значення для забезпечення достовірності результатів аналізу. Використання бібліотек NumPy і SciPy гарантує високий рівень точності завдяки використанню методів з подвійною точністю та обчислювальним стандартам IEEE 754. У контексті моделювання та оцінки ризиків, зокрема при розрахунку метрик,

таких як Value at Risk (VaR) чи Conditional Value at Risk (CVaR), похибки обчислень залишаються мінімальними навіть за умови роботи з великими обсягами даних. Аналіз похибок у результатах показав, що відхилення не перевищує допустимих меж, визначених для подібних економічних задач, що дозволяє зробити висновок про відповідність результатів стандартам галузі.

Важливою складовою оцінки є оптимальність використання ресурсів, включаючи оперативну пам’ять і процесорний час. Архітектура програмного забезпечення розроблена таким чином, щоб мінімізувати споживання ресурсів завдяки використанню ефективних структур даних і алгоритмів. Наприклад, для роботи з великими наборами даних використовується бібліотека Pandas, яка забезпечує ефективну обробку табличних даних і агрегування інформації. У тестових середовищах було встановлено, що програма споживає помірну кількість оперативної пам’яті навіть при роботі з великими масивами історичних даних ринку або симуляціями Монте-Карло. Крім того, виконувані процеси оптимізовані для багатоядерних процесорів, що дозволяє розподіляти навантаження та значно знижувати час виконання складних розрахунків.

Масштабованість програмного забезпечення також була піддана детальному аналізу. Це поняття визначає здатність системи підтримувати ефективну роботу при збільшенні обсягу даних або числа користувачів. У процесі тестування було проведено серію експериментів із поступовим збільшенням обсягів вхідних даних, включаючи сценарії, які імітують роботу у великомасштабних корпоративних середовищах. Результати показали, що час виконання алгоритмів і споживання ресурсів збільшується пропорційно зростанню розмірності задачі, що свідчить про відсутність проблем із масштабованістю. Крім того, використання сучасних бібліотек Python, таких як Dask, дозволяє реалізувати паралельну обробку даних, що ще більше підвищує продуктивність системи у великих середовищах.

Стабільність програмного забезпечення була перевірена за допомогою тестів на різноманітних наборах даних і у різних сценаріях використання.

Тести включали роботу з шумовими даними, пропусками у вхідних масивах, а також аналіз крайових випадків, наприклад, з нульовими або дуже великими значеннями вхідних параметрів. Результати показали, що програма стабільно обробляє навіть найбільш складні набори даних, забезпечуючи коректні результати і попереджаючи користувачів про можливі проблеми з якістю вхідних даних через зрозумілі повідомлення про помилки. Крім того, програмний код було протестовано на різних операційних системах і конфігураціях апаратного забезпечення, що підтвердило його сумісність і відсутність критичних помилок.

Адаптивність до різних сценаріїв використання є ще одним важливим критерієм оцінки ефективності. Система була розроблена з урахуванням можливості її застосування в різних галузях економіки та фінансів, що забезпечується завдяки гнучкості архітектури. Наприклад, модуль оптимізації портфеля може бути адаптований як для завдань управління активами, так і для аналізу ризиків в енергетичному секторі. Модуль аналізу ринкової рівноваги, у свою чергу, може використовуватися як для дослідження споживчого попиту, так і для моделювання взаємодії між підприємствами в конкурентному середовищі. Гнучкість забезпечується за рахунок можливості налаштування вхідних параметрів, що дозволяє користувачам легко адаптувати систему до своїх потреб.

Крім того, оцінювання включало порівняння продуктивності і точності програмного забезпечення з аналогічними рішеннями, доступними на ринку. Результати показали, що розроблений програмний продукт демонструє конкурентну продуктивність і має низку переваг, таких як інтеграція сучасних методів оптимізації та моделювання, а також висока якість візуалізації даних. Наприклад, у порівнянні з комерційними програмами для фінансового аналізу, це програмне забезпечення забезпечує більш детальну кастомізацію і відкритість коду, що дозволяє користувачам модифікувати алгоритми відповідно до своїх потреб.

У підсумку, проведений аналіз ефективності програмного забезпечення свідчить про його високий рівень як з точки зору продуктивності, так і з точки зору адаптивності та гнучкості. Завдяки використанню сучасних технологій і ретельно продуманій архітектурі код забезпечує швидку, точну і стабільну роботу у широкому спектрі застосувань. Цей результат досягається завдяки комбінації ефективних алгоритмів, оптимального використання ресурсів і можливості адаптації до різних умов, що робить програмне забезпечення надійним і перспективним інструментом для вирішення складних економічних задач.

3.7 Висновки до третього розділу

Розділ відображає послідовність етапів створення програмного забезпечення, починаючи від постановки задачі, вибору технологій, опису архітектури і закінчуючи тестуванням та аналізом ефективності. Це вказує на систематичний підхід до розробки. Вибір інструментів і технологій (таких як NumPy, Pandas, Matplotlib, SciPy) відповідає потребам реалізації задачі. Кожен компонент системи був підібраний з урахуванням специфіки поставлених цілей, що свідчить про оптимальність рішень.

Опис процесу роботи розкриває деталі алгоритмів, методів та способів інтеграції компонентів. Це демонструє ретельний підхід до побудови системи та врахування можливих проблем на етапі реалізації.

Опис архітектури програмного забезпечення вказує на модульність системи, яка забезпечує зручність тестування, розширення функціональності та повторного використання коду. Тестування реалізації включає оцінку правильності функціонування, продуктивності та стабільності коду. Це демонструє надійність і відповідність системи заданим вимогам.

ВИСНОВКИ

Дослідження сучасних алгоритмів оптимізації в економіці дозволило отримати важливі висновки щодо ролі, можливостей і обмежень цих методів у контексті сучасних економічних викликів. Оптимізація займає центральне місце в економічній науці, оскільки вона спрямована на ефективне використання обмежених ресурсів, мінімізацію витрат і максимізацію результатів. У сучасному світі, де економіка функціонує в умовах глобалізації, цифровізації та швидких змін, використання новітніх алгоритмів оптимізації стає не лише корисним, а й необхідним.

Дослідження підтвердило, що алгоритми лінійного та нелінійного програмування залишаються фундаментальними інструментами для вирішення широкого спектра задач, таких як планування виробничих процесів, розподіл ресурсів та управління логістикою. Їх точність і відносна простота у використанні роблять їх придатними для багатьох стандартних економічних проблем. Однак в умовах реального світу, де задачі стають дедалі складнішими та часто включають нелінійні взаємозв’язки між змінними, класичні підходи мають обмеження. Тому значна увага приділяється розробці та впровадженню еволюційних алгоритмів, таких як генетичні алгоритми та алгоритми роїв, які здатні ефективно працювати в багатовимірних і динамічних середовищах.

Методи машинного навчання були визначені як одна з найбільш перспективних областей для застосування в оптимізації. Алгоритми глибокого навчання та навчання з підкріпленням демонструють здатність адаптуватися до умов невизначеності та обробляти великі обсяги даних, що є критичним у сучасній економіці. Наприклад, такі алгоритми успішно використовуються у фінансовій оптимізації, для моделювання ризиків і прогнозування ринкових тенденцій. Вони також мають значний потенціал у логістиці, управлінні запасами, ціноутворенні та оптимізації бізнес-процесів.

Розгляд теорії ігор і теорії графів показав їх важливість у моделюванні стратегічних взаємодій між економічними агентами та аналізі структур складних систем. Теорія ігор дозволяє визначати оптимальні стратегії в умовах конкуренції або співпраці, що є особливо важливим для моделювання ринкових взаємодій і розробки політик у сфері управління. Теорія графів, у свою чергу, забезпечує інструменти для аналізу мереж, таких як транспортні або соціальні мережі, та пошуку оптимальних рішень для їх функціонування.

Практичне значення дослідження полягає в тому, що розроблені алгоритми й рекомендації можуть бути застосовані для підвищення ефективності управління економічними процесами, зокрема у виробничих, фінансових і логістичних системах. Впровадження цих алгоритмів сприяє зниженню витрат, оптимізації використання ресурсів і досягненню стратегічних цілей підприємств та організацій.

Загальні висновки дослідження підкреслюють важливість міждисциплінарного підходу до розробки алгоритмів оптимізації, який включає елементи математики, економіки, інформатики та теорії управління. Подальший розвиток цієї галузі сприятиме створенню більш ефективних та адаптивних моделей, здатних відповідати на виклики сучасної економіки.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Y. Sun, J. Shen, X. Zhang, and C. Sun, “A Particle Swarm Optimization with Dynamic Strategy for Multi-Modal Multi-Objective Location Optimization Problem”, in *2023 5th Int. Conf. Data-driven Optim. Complex Syst. (DOCS)*, Tianjin, China, Sep. 22–24, 2023. IEEE, 2023. https://doi.org/10.1109/docs60977.2023.10294853

2. W. Zhao, L. Wang, and Z. Zhang, “Supply-Demand-Based Optimization: A Novel Economics-Inspired Algorithm for Global Optimization”, *IEEE Access*, vol. 7, pp. 73182–73206, 2019. https://doi.org/10.1109/access.2019.2918753

3. N. Wang, J. S. Wang, L. F. Zhu, H. Y. Wang, and G. Wang, “Novel Dynamic Clustering Method by Integrating Marine Predators Algorithm and Particle Swarm Optimization Algorithm”, *IEEE Access*, p. 1, 2020. https://doi.org/10.1109/access.2020.3047819

4. S. Zhou, “Online Recommendation of Digital Trade Resources Based on Gradient Descent Optimization Algorithm”, in *2024 Int. Conf. Data Sci. Netw. Secur. (ICDSNS)*, Tiptur, India, Jul. 26–27, 2024. IEEE, 2024, pp. 1–5. https://doi.org/10.1109/icdsns62112.2024.10691186

5. J. Guo, G. Zhou, Y. Di, B. Shi, K. Yan, and Y. Sato, “A Bare-Bones Particle Swarm Optimization With Crossed Memory for Global Optimization”, *IEEE Access*, vol. 11, pp. 31549–31568, 2023. https://doi.org/10.1109/access.2023.3250228

6. H. Wu, Q. Li, W. Wei, and X. Zeng, “A Study on Improved Particle Swarm Optimization Algorithm in Portfolio Optimization Problem”, in *2023 China Automat. Congr. (CAC)*, Chongqing, China, Nov. 17–19, 2023. IEEE, 2023. https://doi.org/10.1109/cac59555.2023.10451733

7. F. Hasan, F. Ahmad, M. Shahid, A. Khan, and G. Ahmad, “Solving Portfolio Selection Problem Using Whale Optimization Algorithm”, in *2022 3rd Int.*

Conf. Computation, Automat. Knowl. Manage. (ICCAKM), Dubai, United Arab Emirates, Nov. 15–17, 2022. IEEE, 2022. https://doi.org/10.1109/iccakm54721.2022.9990079

8. I. Huseyinov and S. Ulucay, “Application of Genetic and Particle Swarm Optimization Algorithms to Portfolio Optimization Problem: Borsa İstanbul and Crypto Money Exchange”, in *2019 4th Int. Conf. Comput. Sci. Eng. (UBMK)*, Samsun, Turkey, Sep. 11–15, 2019. IEEE, 2019. https://doi.org/10.1109/ubmk.2019.8907225

9. P. K. Aithal, M. Geetha, U. Dinesh Acharya, B. Savitha, and P. Menon, “Real-Time Portfolio Management System Utilizing Machine Learning Techniques”, *IEEE Access*, p. 1, 2023. https://doi.org/10.1109/access.2023.3263260

10. C.-a. Liu and T. Jiang, “Smooth Multiobjective Portfolio Optimization Model and Its Solving Method”, in *2021 Int. Conf. Inf. Technol. Biomed. Eng. (ICITBE)*, Nanchang, China, Dec. 24–26, 2021. IEEE, 2021. https://doi.org/10.1109/icitbe54178.2021.00048

11. A. Abbas and K. Raza, “Sharpe Index Based Portfolio Optimization Using Computational Intelligence”, in *2023 IEEE Int. Conf. Artif. Intell. Eng. Technol. (IICAIET)*, Kota Kinabalu, Malaysia, Sep. 12–14, 2023. IEEE, 2023. https://doi.org/10.1109/iicaiet59451.2023.10291936

12. M. Altinoz and O. T. Altinoz, “Systematic Initialization Approaches for Portfolio Optimization Problems”, *IEEE Access*, vol. 7, pp. 57779–57794, 2019. https://doi.org/10.1109/access.2019.2914115

13. Z. X. Loke, S. L. Goh, and J. Likoh, “A Self-adaptive Step-size Search Algorithm for the Cardinality Constrained Portfolio Optimisation Problem”, in *2024 20th IEEE Int. Colloq. Signal Process. & Its Appl. (CSPA)*, Langkawi, Malaysia, Mar. 1–2, 2024. IEEE, 2024. https://doi.org/10.1109/cspa60979.2024.10525478

14. K. Erwin and A. Engelbrecht, “Improved Set-based Particle Swarm optimization for Portfolio optimization”, in *2020 IEEE Symp. Ser. Comput. Intell.*

(SSCI), Canberra, Australia, Dec. 1–4, 2020. IEEE, 2020. https://doi.org/10.1109/ssci47803.2020.9308579

15. D. M. Hoang, T. N. Thang, N. D. Tu, and N. V. Hoang, "Stochastic Linear Programming Approach for Portfolio Optimization Problem", in *2021 IEEE Int. Conf. Mach. Learn. Appl. Netw. Technol. (ICMLANT)*, Soyapango, El Salvador, Dec. 16–17, 2021. IEEE, 2021. https://doi.org/10.1109/icmlant53170.2021.9690552

16. M.-F. Leung and J. Wang, "Minimax and Biobjective Portfolio Selection Based on Collaborative Neurodynamic Optimization", *IEEE Trans. Neural Netw. Learn. Syst.*, pp. 1–12, 2020. https://doi.org/10.1109/tnnls.2019.2957105

ДОДАТКИ

Додаток А

Лістинг програмного коду

```
import numpy as np
import scipy.optimize as optimize
import matplotlib.pyplot as plt
import pandas as pd
from typing import List, Tuple, Callable, Dict, Optional
from dataclasses import dataclass
from datetime import datetime
import seaborn as sns
from scipy.stats import norm
import warnings
warnings.filterwarnings('ignore')

@dataclass
class OptimizationResult:
    """Структура для зберігання результатів оптимізації"""
    success: bool
    solution: np.ndarray
    objective_value: float
    iterations: int
    message: str
    timestamp: datetime = datetime.now()

class AdvancedEconomicOptimizer:
    """
    Розширений клас для економічної оптимізації з додатковими функціями
    та методами аналізу
    """

    def __init__(self, random_seed: int = 42):
        """
        Ініціалізація оптимізатора

        Args:
            random_seed (int): Зерно для генератора випадкових чисел
        """
        np.random.seed(random_seed)
```

```
    self.optimization_history: List[OptimizationResult] = []

def save_result(self, result: OptimizationResult):
    """Збереження результату оптимізації в історію"""
    self.optimization_history.append(result)

def get_optimization_history(self) -> pd.DataFrame:
    """Повертає історію оптимізації як DataFrame"""
    return pd.DataFrame([
        {
            'timestamp': r.timestamp,
            'success': r.success,
            'objective_value': r.objective_value,
            'iterations': r.iterations,
            'message': r.message
        }
        for r in self.optimization_history
    ])

def linear_programming(
    self,
    c: np.ndarray,
    A_ub: np.ndarray,
    b_ub: np.ndarray,
    A_eq: Optional[np.ndarray] = None,
    b_eq: Optional[np.ndarray] = None
) -> OptimizationResult:
    """
    Розширений метод лінійного програмування

    Args:
        c: Коефіцієнти цільової функції
        A_ub: Матриця нерівностей обмежень
        b_ub: Вектор правих частин нерівностей
        A_eq: Матриця рівностей обмежень
        b_eq: Вектор правих частин рівностей
    """
    result = optimize.linprog(
        c, A_ub=A_ub, b_ub=b_ub,
        A_eq=A_eq, b_eq=b_eq,
        method='highs'
    )
```

```
        optimization_result = OptimizationResult(
            success=result.success,
            solution=result.x,
            objective_value=result.fun,
            iterations=result.nit,
            message=result.message
        )

        self.save_result(optimization_result)
        return optimization_result

    def stochastic_portfolio_optimization(
        self,
        returns: np.ndarray,
        n_scenarios: int = 1000,
        risk_free_rate: float = 0.02,
        confidence_level: float = 0.95
    ) -> Tuple[OptimizationResult, Dict]:
        """
        Стохастична оптимізація портфеля з використанням методу Монте-Карло

        Args:
            returns: Історичні дохідності активів
            n_scenarios: Кількість сценаріїв для симуляції
            risk_free_rate: Безризикова ставка
            confidence_level: Рівень довіри для VaR
        """
        n_assets = returns.shape[1]
        mean_returns = np.mean(returns, axis=0)
        cov_matrix = np.cov(returns.T)


        scenarios = np.random.multivariate_normal(
            mean_returns,
            cov_matrix,
            n_scenarios
        )

        def portfolio_metrics(weights):
            portfolio_returns = np.dot(scenarios, weights)
            portfolio_mean = np.mean(portfolio_returns)
            portfolio_std = np.std(portfolio_returns)
```

```
            sharpe = (portfolio_mean - risk_free_rate) / portfolio_std
            var = np.percentile(portfolio_returns, (1 - confidence_level) * 100)
            cvar = np.mean(portfolio_returns[portfolio_returns <= var])
            return -sharpe, var, cvar

        constraints = [
            {'type': 'eq', 'fun': lambda x: np.sum(x) - 1},
            {'type': 'ineq', 'fun': lambda x: x}
        ]

        result = optimize.minimize(
            lambda w: portfolio_metrics(w)[0],
            x0=np.ones(n_assets) / n_assets,
            method='SLSQP',
            constraints=constraints
        )

        final_sharpe, var, cvar = portfolio_metrics(result.x)

        metrics = {
            'sharpe_ratio': -final_sharpe,
            'var': var,
            'cvar': cvar,
            'expected_return': np.dot(mean_returns, result.x),
            'portfolio_volatility': np.sqrt(np.dot(result.x.T, np.dot(cov_matrix,
result.x)))
        }

        optimization_result = OptimizationResult(
            success=result.success,
            solution=result.x,
            objective_value=result.fun,
            iterations=result.nit,
            message=result.message
        )

        self.save_result(optimization_result)
        return optimization_result, metrics

    def dynamic_production_optimization(
        self,
        demand_forecast: np.ndarray,
        production_costs: np.ndarray,
```

```
    storage_costs: np.ndarray,
    initial_inventory: float,
    max_production: float,
    max_storage: float
) -> OptimizationResult:
    """
    Динамічна оптимізація виробництва з урахуванням прогнозу попиту

    Args:
        demand_forecast: Прогноз попиту по періодах
        production_costs: Витрати на виробництво по періодах
        storage_costs: Витрати на зберігання по періодах
        initial_inventory: Початковий запас
        max_production: Максимальний обсяг виробництва
        max_storage: Максимальний обсяг зберігання
    """
    n_periods = len(demand_forecast)

    def objective(x):

        production = x[:n_periods]
        inventory = x[n_periods:]


        total_cost = np.sum(production * production_costs +
                            inventory * storage_costs)

        return total_cost

    def constraints(x):
        production = x[:n_periods]
        inventory = x[n_periods:]


        constraints = []
        current_inventory = initial_inventory

        for t in range(n_periods):
            current_inventory += production[t] - demand_forecast[t]
            constraints.append(current_inventory - inventory[t])
            current_inventory = inventory[t]

        return np.array(constraints)
```

```
    x0 = np.zeros(2 * n_periods)
    x0[:n_periods] = demand_forecast


    bounds = [(0, max_production)] * n_periods + [(0, max_storage)] * n_periods
    constraint = {'type': 'eq', 'fun': constraints}

    result = optimize.minimize(
        objective,
        x0,
        method='SLSQP',
        bounds=bounds,
        constraints=constraint
    )

    optimization_result = OptimizationResult(
        success=result.success,
        solution=result.x,
        objective_value=result.fun,
        iterations=result.nit,
        message=result.message
    )

    self.save_result(optimization_result)
    return optimization_result

def market_equilibrium_with_elasticity(
    self,
    base_price: float,
    base_quantity: float,
    supply_elasticity: float,
    demand_elasticity: float,
    market_shocks: Optional[Dict[str, float]] = None
) -> Tuple[float, float, Dict]:
    """
    Розрахунок ринкової рівноваги з урахуванням еластичності та шоків

    Args:
        base_price: Базова ціна
        base_quantity: Базова кількість
        supply_elasticity: Еластичність пропозиції
```

```
    demand_elasticity: Еластичність попиту
    market_shocks: Словник з шоками ринку
"""
if market_shocks is None:
    market_shocks = {}

def supply_curve(p):
    quantity = base_quantity * (p / base_price) ** supply_elasticity
    for shock, magnitude in market_shocks.get('supply', {}).items():
        quantity *= (1 + magnitude)
    return quantity

def demand_curve(p):
    quantity = base_quantity * (p / base_price) ** (-demand_elasticity)
    for shock, magnitude in market_shocks.get('demand', {}).items():
        quantity *= (1 + magnitude)
    return quantity

def market_clearing(p):
    return supply_curve(p) - demand_curve(p)

result = optimize.root_scalar(
    market_clearing,
    bracket=[base_price * 0.1, base_price * 10],
    method='brentq'
)

equilibrium_price = result.root
equilibrium_quantity = supply_curve(equilibrium_price)


consumer_surplus = self._calculate_consumer_surplus(
    equilibrium_price,
    equilibrium_quantity,
    demand_curve
)

producer_surplus = self._calculate_producer_surplus(
    equilibrium_price,
    equilibrium_quantity,
    supply_curve
)
```

```
    metrics = {
        'consumer_surplus': consumer_surplus,
        'producer_surplus': producer_surplus,
        'total_welfare': consumer_surplus + producer_surplus,
        'price_elasticity_of_demand': demand_elasticity,
        'price_elasticity_of_supply': supply_elasticity
    }

    return equilibrium_price, equilibrium_quantity, metrics

def _calculate_consumer_surplus(
    self,
    price: float,
    quantity: float,
    demand_curve: Callable
) -> float:
    """Розрахунок надлишку споживача"""
    x = np.linspace(price, price * 2, 1000)
    y = demand_curve(x)
    return np.trapz(y, x) - price * quantity

def _calculate_producer_surplus(
    self,
    price: float,
    quantity: float,
    supply_curve: Callable
) -> float:
    """Розрахунок надлишку виробника"""
    x = np.linspace(price * 0.5, price, 1000)
    y = supply_curve(x)
    return price * quantity - np.trapz(y, x)

def risk_analysis(
    self,
    returns: np.ndarray,
    weights: np.ndarray,
    confidence_level: float = 0.95,
    n_simulations: int = 10000
) -> Dict:
    """
    Розширений аналіз ризиків портфеля

    Args:
```

```
        returns: Історичні дохідності
        weights: Ваги активів
        confidence_level: Рівень довіри
        n_simulations: Кількість симуляцій
    """
    portfolio_returns = np.dot(returns, weights)


    mean_return = np.mean(portfolio_returns)
    std_dev = np.std(portfolio_returns)


    var = np.percentile(portfolio_returns, (1 - confidence_level) * 100)


    cvar = np.mean(portfolio_returns[portfolio_returns <= var])


    sim_returns = np.random.normal(
        mean_return,
        std_dev,
        n_simulations
    )


    downside_returns = portfolio_returns[portfolio_returns < 0]
    downside_deviation = np.std(downside_returns) if len(downside_returns) > 0
else 0

    sortino_ratio = (mean_return / downside_deviation) if downside_deviation !=
0 else np.inf


    cumulative_returns = np.cumprod(1 + portfolio_returns)
    running_max = np.maximum.accumulate(cumulative_returns)
    drawdowns = (cumulative_returns - running_max) / running_max
    max_drawdown = np.min(drawdowns)

    return {
        'mean_return': mean_return,
        'volatility': std_dev,
        'var': var,
        'cvar': cvar,
```

```
        'downside_deviation': downside_deviation,
        'sortino_ratio': sortino_ratio,
        'max_drawdown': max_drawdown,
        'skewness': pd.Series(portfolio_returns).skew(),
        'kurtosis': pd.Series(portfolio_returns).kurtosis(),
        'positive_periods': np.sum(portfolio_returns > 0) / len(portfolio_returns),
        'simulation_results': sim_returns
    }

def visualize_optimization_results(
    self,
    results_dict: Dict,
    plot_type: str = 'all'
):
    """
    Візуалізація результатів оптимізації

    Args:
        results_dict: Словник з результатами
        plot_type: Тип візуалізації ('all', 'risk', 'returns', 'distribution')
    """


    if plot_type in ['all', 'distribution']:
        plt.figure(figsize=(12, 6))

        if 'simulation_results' in results_dict:
            sns.histplot(
                results_dict['simulation_results'],
                stat='density',
                kde=True,
                label='Розподіл дохідності'
            )


            if 'var' in results_dict:
                plt.axvline(
                    results_dict['var'],
                    color='red',
                    linestyle='--',
                    label=f"VaR ({results_dict['var']:.2%})"
                )
            if 'cvar' in results_dict:
```

```
                plt.axvline(
                    results_dict['cvar'],
                    color='darkred',
                    linestyle=':',
                    label=f"CVaR ({results_dict['cvar']:.2%})"
                )

            plt.title('Розподіл дохідності портфеля')
            plt.xlabel('Дохідність')
            plt.ylabel('Щільність')
            plt.legend()

        if plot_type in ['all', 'risk']:
            plt.figure(figsize=(10, 6))
            risk_metrics = {
                k: v for k, v in results_dict.items()
                if k in ['volatility', 'downside_deviation', 'var', 'cvar', 'max_drawdown']
            }

            plt.bar(
                risk_metrics.keys(),
                [abs(v) for v in risk_metrics.values()]
            )
            plt.title('Метрики ризику')
            plt.xticks(rotation=45)
            plt.ylabel('Значення')

        if plot_type in ['all', 'returns']:
            plt.figure(figsize=(10, 6))

            if 'simulation_results' in results_dict:
                cumulative_returns = np.cumprod(
                    1 + results_dict['simulation_results']
                )
                plt.plot(cumulative_returns, alpha=0.5)
                plt.title('Кумулятивна дохідність (симуляція)')
                plt.xlabel('Час')
                plt.ylabel('Кумулятивна дохідність')

        plt.tight_layout()
        plt.show()

def demo_advanced_optimization():
```

```
"""Демонстрація розширених можливостей оптимізації"""
optimizer = AdvancedEconomicOptimizer(random_seed=42)


print("\n1. Стохастична оптимізація портфеля:")
n_assets = 5
n_periods = 1000
returns = np.random.normal(0.001, 0.02, (n_periods, n_assets))

result, metrics = optimizer.stochastic_portfolio_optimization(
    returns,
    n_scenarios=1000,
    risk_free_rate=0.02,
    confidence_level=0.95
)

print("Оптимальні ваги:", result.solution)
print("Метрики портфеля:", metrics)


print("\n2. Динамічна оптимізація виробництва:")
n_periods = 12
demand_forecast = 100 + 10 * np.sin(np.linspace(0, 2*np.pi, n_periods))
production_costs = np.ones(n_periods) * 10
storage_costs = np.ones(n_periods) * 2

prod_result = optimizer.dynamic_production_optimization(
    demand_forecast=demand_forecast,
    production_costs=production_costs,
    storage_costs=storage_costs,
    initial_inventory=50,
    max_production=150,
    max_storage=200
)

print("Оптимальний план виробництва:", prod_result.solution[:n_periods])
print("Оптимальний план зберігання:", prod_result.solution[n_periods:])


print("\n3. Аналіз ринкової рівноваги з шоками:")
market_shocks = {
    'supply': {'weather': -0.1, 'technology': 0.15},
    'demand': {'income': 0.2, 'preferences': -0.05}
```

```
    }

    eq_price, eq_quantity, market_metrics =
optimizer.market_equilibrium_with_elasticity(
        base_price=100,
        base_quantity=1000,
        supply_elasticity=0.7,
        demand_elasticity=1.2,
        market_shocks=market_shocks
    )

    print("Рівноважна ціна:", eq_price)
    print("Рівноважна кількість:", eq_quantity)
    print("Ринкові метрики:", market_metrics)


    print("\n4. Розширений аналіз ризиків:")
    weights = result.solution
    risk_metrics = optimizer.risk_analysis(
        returns=returns,
        weights=weights,
        confidence_level=0.95,
        n_simulations=10000
    )

    print("Метрики ризику:", {k: v for k, v in risk_metrics.items() if k !=
'simulation_results'})


    print("\n5. Візуалізація результатів:")
    optimizer.visualize_optimization_results(risk_metrics, plot_type='all')


    print("\n6. Історія оптимізації:")
    history_df = optimizer.get_optimization_history()
    print(history_df)

if __name__ == "__main__":
    demo_advanced_optimization()
```