МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний університет «Запорізька політехніка»

Комп'ютерних наук і технологій

(повне найменування факультету)

Системний аналіз та обчислювальна математика

(повне найменування кафедри)

Пояснювальна записка

до дипломної роботи

магістра

(ступінь вищої освіти)

на тему «Системний аналіз та оптимізація виробничого планування

(назва теми)

підприємства в умовах ресурсних обмежень»

Виконав: студент 4 курсу, групи КНТ-812

Спеціальності 124 Системний аналіз

(код і найменування спеціальності)

Освітня програма (спеціалізація)

«Інтелектуальні технології та прийняття рішень в складних системах»

БУДУРОВ.М.М

(ПРІЗВИЩЕ та ініціали)

Керівник ТЕРЕЩЕНКО Е.В.

(ПРІЗВИЩЕ та ініціали)

Рецензент ЛОЗОВСЬКА Л.І.

(ПРІЗВИЩЕ та ініціали)

2024

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Інститут, факультет КОМП'ЮТЕРНИХ НАУК ТА ТЕХНОЛОГІЙ
Кафедра «Системний аналіз та обчислювальна математика»
Ступінь вищої освіти Бакалавра
Спеціальність 124-Системний аналіз
(код і найменування)
Освітня програма (спеціалізація) «Інтелектуальні технології та прийняття рішень в складних системах»
(назва освітньої програми (спеціалізації)

ЗАТВЕРДЖУЮ
Завідувач кафедри САОМ
Еліна ТЕРЕЩЕНКО
08 червня 2026 року

ЗАВДАННЯ
НА ДИПЛОМНУ РОБОТУ СТУДЕНТА

БУДУРОВА Микити Максимовича
(ПРІЗВИЩЕ, ім'я, по батькові)

1. Тема роботи «Системний аналіз та оптимізація виробничого планування підприємства в умовах ресурсних обмежень»
к.ф.-м.н, доцент, ТЕРЕЩЕНКО Еліна Валентинівна
(науковий ступінь, вчене звання, ПРІЗВИЩЕ, ім'я, по батькові)
затверджені наказом закладу вищої освіти від «23»квітня 2026 року №196
2. Строк подання студентом роботи «04» червня 2026 року
3. Вихідні дані до роботи

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) .
1.Аналіз діяльності підприємства та ідентифікація його виробничі потреби.
2. Формалізація математичної моделі розрахунку тіньових цін.
3. Розробка архітектури системи підтримки прийняття рішень
4. Комп'ютерне моделювання та сценарний аналіз стійкості рішень.

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

6. Консультанти розділів проекту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Прийняв виконане завдання
Розділ 1, Розділ 2	Терещенко Е.В., зав. кафедри	04.05.2026	03.06.2026
Нормоконтроль	Широкорад Д.В., доцент	05.06.2026	08.06.2026

7. Дата видачі завдання «06» березня 2026 року

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів проекту (роботи)	Примітка
1	Постановка завдання роботи.	1 тиждень	Вступ
2	Проведення аналізу предметної області.	1 тиждень	Розділ 1
3	Збір, аналіз та систематизація вихідних даних.	1 тиждень	Розділ 2
4	Обґрунтування вибору інструментарію розробки.	2 тиждень	Розділ 1, 2
5	Проектування логічної архітектури.	3 тиждень	Розділ 3
6	Програмна реалізація обчислювального ядра	3-4 тиждень	Розділ 3, Додаток А
7	Проведення серії експериментів	5 тиждень	Розділ 4
8	Оформлення текстової пояснювально-розрахункової записки згідно з вимогами нормоконтролю.	6 тиждень	Пояснювальна записка
9	Подання роботи на нормоконтроль та рецензування.	6 тиждень	Відгук, рецензія
10	Підготовка для публічного захисту перед екзаменаційною комісією.	7-8 тиждень	Презентація, доповідь

Студент _____ БУДУРОВ М.М.
(підпис) (прізвище та ініціали)

Керівник роботи _____ ТЕРЕЩЕНКО Е.В.
(підпис) (прізвище та ініціали)

РЕФЕРАТ

ПЗ: 55с., 13 рис., 3 табл., 20 джерел, 1 додаток.

Об'єктом дослідження виступають процеси операційного планування асортименту продукції на підприємстві.

Предметом дослідження є математичні методи ціло-чисельного лінійного програмування, теорія двоїстості та алгоритми багатовимірного аналізу стійкості складних економічних систем.

Метою роботи є підвищення фінансово-економічної ефективності та стійкості функціонування фабрики шляхом побудови гнучких оптимізаційних моделей і розробки прикладної комп'ютерної системи для операційного менеджменту.

Для розв'язання сформованої оптимізаційної задачі в загальному вигляді використано гібридний математичний підхід «солвера». Спочатку застосовується лінійний симплекс-метод, що дозволяє знайти неперервну релаксацію моделі та обчислити вектори двоїстих оцінок для визначення прихованої цінності ресурсів.

Практична цінність дослідження полягає у створенні автономного програмного комплексу на базі універсальної мови програмування Python з використанням бібліотек PuLP, Pandas та Matplotlib. Програма інтегрована з реляційною СУБД SQLite та оснащена нативним графічним інтерфейсом користувачаз з вбудованими модулями валідації даних.

За допомогою автоматично згенерованих двовимірних графіків аналізу чутливості та тривимірних об'ємних поверхонь відгуку, побудованих за допомогою регулярних сіток координат, наочно продемонстровано ресурсну конкуренцію та визначено критичні точки зламу траєкторії прибутку.

СИСТЕМНИЙ АНАЛІЗ ТА ОПТИМІЗАЦІЯ ВИРОБНИЧОГО ПЛАНУВАННЯ ПІДПРИЄМСТВА В УМОВАХ РЕСУРСНИХ ОБМЕЖЕНЬ.

ЗМІСТ

ВСТУП

У сучасних умовах функціонування промислових підприємств ефективне управління ресурсами стає визначальним фактором конкурентоспроможності[1]. Обмеженість сировини, криза енергетичного ринку та дефіцит кваліфікованої робочої сили вимагають від менеджменту впровадження високоточних методів планування. Системний аналіз дозволяє розглядати підприємство як комплексну структуру, де зміна одного параметра впливає на всю результативність системи. Використання методів математичного моделювання для пошуку оптимального балансу між витратами та прибутком є критично важливим для сталого розвитку бізнесу.

Традиційні методи планування часто не враховують динамічну зміну зовнішніх факторів, таких як ліміти енергоспоживання або раптові зміни в поставках матеріалів. Це призводить до неефективного використання потужностей або невиконання мінімальних планових зобов’язань.

Об’єкт дослідження — процес виробничого планування на підприємстві в умовах жорстких ресурсних обмежень на прикладі меблевого виробництва.

Предмет дослідження — математичні моделі та програмні інструменти системного аналізу, зокрема методи лінійного програмування та їх реалізація засобами мови Python.

Мета роботи — полягає у розробці та впровадженні автоматизованої системи оптимізації виробничого плану, яка дозволяє максимізувати прибуток підприємства, враховуючи обмеження на кількість працівників, людино-годин, електроенергії та сировини.

Для досягнення мети було поставлено наступні завдання:

1. Проаналізувати структуру виробничих процесів та визначити ключові ресурсні обмеження.
2. Розробити реляційну структуру бази даних для зберігання техніко-економічних показників виробництва.

3. Побудувати математичну модель максимізації цільової функції (прибутку) за допомогою бібліотеки PuLP.

4. Реалізувати алгоритм імпорту та обробки даних за допомогою Pandas та SQLite.

5. Провести аналіз чутливості моделі для оцінки стійкості виробничого плану при зміні лімітів електроенергії та візуалізувати результати.

Методи дослідження. У роботі застосовано апарат лінійного програмування, методи системного аналізу, теорію баз даних та сучасні бібліотеки аналізу даних на мові Python.

Практична цінність — Розроблений програмний комплекс дозволяє швидко перераховувати виробничі плани при зміні будь-яких вхідних параметрів, забезпечуючи прийняття обґрунтованих управлінських рішень у режимі реального часу.

РОЗДІЛ 1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Функціонування меблевого підприємства як складної економічної системи

З позицій системного аналізу виробниче планування є складною багатокритеріальною системою. Сучасне промислове підприємство, що спеціалізується на виготовленні меблевої продукції, являє собою складну динамічну систему, функціонування якої залежить від ефективної взаємодії технологічних, економічних та логістичних процесів. У межах системного аналізу таке підприємство розглядається як механізм трансформації вхідних ресурсів у готову продукцію з метою отримання доданої вартості та максимізації прибутку[1].

Об’єктом дослідження у цій роботі виступає виробничий комплекс, асортиментна матриця якого включає три базові категорії товарів: стільці, столи та шафи. Кожна з цих категорій має унікальні технологічні характеристики, різний рівень матеріаломісткості та енергоємності, що створює передумови для багатоваріантності планування.

Розвиток меблевої галузі в сучасних умовах характеризується високим рівнем конкуренції та значною залежністю від вартості енергоносіїв і сировини. Також цифрова трансформація процесів планування стає невід'ємною частиною ефективного управління [2]. Для системного аналітика ключовим завданням є не просто опис виробничого циклу, а ідентифікація внутрішніх і зовнішніх факторів, що перешкоджають досягненню глобального оптимуму системи[3]. Це вимагає детального вивчення структури витрат, де основну частку займають пиломатеріали, електроенергія та оплата праці кваліфікованого персоналу. У свою чергу економічна стійкість меблевих підприємств безпосередньо залежить від раціонального використання лімітованих матеріалів [4].»

Еволюція системного аналізу в управлінні промисловим виробництвом:

Історія системного аналізу як наукової дисципліни тісно переплетена з розвитком індустріального суспільства та необхідністю вирішення складних управлінських задач. На початкових етапах розвитку промисловості, у період панування ремісничого виробництва, управління базувалося виключно на інтуїції та індивідуальному досвіді власника. Проте з початком першої промислової революції та появою мануфактур виникла гостра потреба в раціоналізації праці. Перші спроби системного підходу можна простежити в працях Фредеріка Тейлора, який запропонував концепцію наукового менеджменту. Його підхід полягав у декомпозиції складних виробничих процесів на елементарні операції, що дозволяло аналізувати кожну ланку системи окремо та оптимізувати її часові та ресурсні витрати.

Подальший розвиток системного аналізу був зумовлений появою конвеєрного виробництва Генрі Форда, де взаємозалежність між окремими елементами системи стала критичною. Будь-який збій в одній ланці призводив до зупинки всього підприємства, що змусило дослідників шукати методи координації та синхронізації потоків сировини та енергії. Однак справжній прорив відбувся під час Другої світової війни, коли виникла потреба в ефективному розподілі обмежених військових ресурсів. Саме тоді сформувався напрям «Дослідження операцій» Operations Research, а в 1947 році Джордж Данціг представив симплекс-метод — алгоритм, який і сьогодні залишається фундаментом для вирішення задач лінійного програмування, подібних до тих, що розглядаються у даній дипломній роботі.

У другій половині XX століття, з поширенням обчислювальної техніки, системний аналіз трансформувався з набору математичних формул у комплексну методологію проектування інформаційних систем управління. З’явилися концепції Material Requirements Planning та пізніше Enterprise Resource Planning, які намагалися інтегрувати всі ресурси підприємства в єдину цифрову модель. У сучасних умовах, які часто характеризують як Індустрію 4.0, системний аналіз меблевого підприємства виходить за межі простого розрахунку кількості матеріалів. Він включає в себе врахування нестабільність

енергетичного ринку, екологічних стандартів та швидкої зміни споживчих уподобань. Сучасний системний аналітик розглядає підприємство не як статичний об'єкт, а як динамічний організм, що постійно адаптується до зовнішнього середовища, де оптимізація виробничого плану є життєво необхідним механізмом виживання в умовах дефіциту енергоресурсів та сировини.

1.2 Ідентифікація та характеристика ключових ресурсних обмежень

Ефективність виробничого планування безпосередньо залежить від точності визначення лімітів, що обмежують можливості системи. У контексті даного дослідження розглядаються чотири критичні групи параметрів обмежень, які визначають межі допустимих рішень. Першою групою є матеріальні ресурси, зокрема деревина, обсяг якої лімітований можливостями складських приміщень та графіками постачання, на розрахунковий період. Другою, і найбільш непередбачуваною групою, є енергетичні ресурси. Електроенергія виступає жорстким лімітом, перевищення якого в умовах дефіциту потужностей може призвести до зупинки виробництва або критичних штрафних санкцій.

Трудовий потенціал підприємства формує третю групу обмежень. Наявність штату працівників зумовлює загальний фонд робочого часу, що є фізичною межею виробничої потужності без залучення додаткових змін або аутсорсингу. Четвертим типом обмежень є ринкові та договірні зобов'язання, які формалізуються у вигляді мінімально необхідної кількості продукції.

Підприємство зобов'язане виготовити певну кількість стільців, столів та шаф для забезпечення стабільності збуту та виконання попередніх замовлень. Системний підхід вимагає, щоб кожен вид продукції споживав певну частку з усіх вищезгаданих категорій обмежень, створюючи складну мережу взаємозалежностей. Зокрема, значний вплив на виробництво має енергетична волатильність та обмеженість енергоносіїв [4].

1.3 Теоретико-методологічні аспекти оптимізації виробничого плану

Для вирішення задачі розподілу обмежених ресурсів між конкурентними напрямками виробництва доцільно використовувати методи математичного програмування. Теоретичною базою для оптимізації слугують фундаментальні методи дослідження операцій. Системний підхід пропонує використання лінійних моделей у випадках, коли прибуток та витрати ресурсів мають пряму пропорційну залежність від обсягу випуску.

Основним інструментом оптимізації в роботі обрано симплекс-метод, який дозволяє ітераційно рухатися по межах області допустимих рішень до знаходження точки максимуму цільової функції.

Важливою частиною методології є проведення аналізу чутливості системи. У реальних умовах параметри обмежень, особливо ліміти електроенергії, не є статичними. Системний аналіз дозволяє оцінити, наскільки стійким є отриманий оптимальний план до зовнішніх збурень. Це у свою чергу дозволяє формалізувати економічні процеси у вигляді строгих математичних рівнянь [4]. Це дає можливість керівництву підприємства не просто отримувати статичну цифру плану, а розуміти граничні межі ефективності системи при зміні вхідних параметрів, що є критично важливим для стратегічного планування.

1.4 Обґрунтування вибору інструментарію для програмної реалізації

У межах системного аналізу складних економічних систем критично важливим етапом є вибір інструментального середовища для моделювання та розробки прикладного програмного забезпечення. Ефективність

функціонування системи підтримки прийняття рішень безпосередньо залежить від гнучкості, масштабованості та інтеграційних можливостей обраного технологічного стеку [5]. Під час проектування даного програмного комплексу було проведено порівняльний аналіз трьох найбільш поширених платформ для розв’язання задач математичної оптимізації, а саме табличного процесора Microsoft Excel, спеціалізованого інженерного середовища MATLAB та універсальної мови програмування Python. Результати цього аналізу довели доцільність використання мови Python як базового інструменту розробки.

Традиційним та найбільш доступним засобом для розв’язання задач лінійного програмування на підприємствах є табличний процесор Microsoft Excel із вбудованою надбудовою «Пошук рішення» Solver [6]. Проте використання Excel для побудови повноцінної системи підтримки прийняття рішень має суттєві обмеження, що унеможливлюють його застосування в межах даного дослідження. Головним недоліком табличного підходу є жорстка прив'язка математичної моделі до конкретної координатної сітки комірок, що робить структуру моделі статичною та вразливою до випадкових змін користувача.[7] Автоматизація динамічного завантаження даних із зовнішніх реляційних баз даних в Excel потребує написання макросів мовою VBA, яка на сьогодні є морально застарілою, небезпечною та складною в супроводі[8]. Крім того, Excel має жорсткі обмеження на кількість змінних і обмежень у базовій версії солвера, не підтримує повноцінне об'єктно-орієнтоване проектування, унеможливлює розділення інтерфейсу та обчислювальної логіки за шаблоном MVC, а бінарний формат файлів Excel робить практично неможливим використання сучасних систем контролю версій типу Git для командної розробки та аудиту коду [9].

Спеціалізований матричний комплекс MATLAB є потужним науковим інструментом, що володіє розвиненими математичними бібліотеками зокрема Optimization Toolbox для вирішення складних задач дослідження операцій. Однак MATLAB є комерційним закритим програмним продуктом із надзвичайно дорогою ліцензією, що суперечить сучасній концепції

впровадження бюджетних та відкритих ІТ-рішень на вітчизняних підприємствах [10]. З інженерної точки зору, MATLAB є громіздким середовищем, яке потребує значних системних ресурсів для розгортання. Створення автономних програм з власним інтерфейсом у MATLAB є ускладненим і потребує додаткового інструментарію MATLAB Compiler, який генерує важкі для запуску екзекуційні файли. Платформа також має обмежені та менш гнучкі засоби для створення, графічних інтерфейсів користувача та прямої низькорівневої інтеграції з реляційними базами даних типу SQLite порівняно із сучасними загальнодоступними мовами програмування [11].

На противагу розглянутим альтернативам, мова програмування Python поєднує в собі переваги потужного математичного процесора та універсального інструменту програмної інженерії. Та застосування алгоритмів цілочисельної оптимізації дозволяє автоматизувати прийняття рішень на виробництві. Будучи повністю безкоштовним продуктом з відкритим вихідним кодом, Python забезпечує повну юридичну та фінансову незалежність розробленого рішення для підприємства. Завдяки наявності високорівневої бібліотеки PuLP, Python дозволяє описувати математичні моделі лінійного програмування у формі, максимально наближеній до теоретичних алгебраїчних рівнянь, що мінімізує ймовірність помилок при трансляції моделі в код. Бібліотека Pandas забезпечує векторизовану, високошвидкісну обробку вхідних масивів даних, а вбудований модуль sqlite3 дозволяє організувати пряму роботу з реляційними структурами даних без залучення важких серверних СУБД.

Ключовою перевагою Python у межах цієї роботи є можливість створення легкої, цілісної архітектури додатку, де в межах одного скрипту за допомогою бібліотеки Tkinter реалізовано графічний інтерфейс користувача, за допомогою PuLP — математичний солвер, а за допомогою Matplotlib — складні аналітичні 2D та 3D графіки. Таке програмне рішення є автономним, легко масштабується під будь-яку кількість нових продуктів чи ресурсів підприємства шляхом звичайної модифікації бази даних, легко піддається версіонуванню в Git та може бути скомпільоване у компактний виконуваний

файл, готовий до безпосереднього інтегрування в операційну діяльність менеджменту[12].

РОЗДІЛ 2 МАТЕМАТИЧНЕ МОДЕЛЮВАННЯ СИСТЕМИ

2.1 Побудова математичної моделі функціонування підприємства

З позицій системного аналізу процес оптимізації виробничого планування формалізується у вигляді загальної задачі лінійного програмування з додатковими умовами цілочисельності для керованих змінних. Математична модель у загальному вигляді описує взаємодію між вектором керованих параметрів, матрицею технологічних коефіцієнтів та вектором обмежень зовнішнього середовища.

Введемо вектор керованих змінних $X = (x_1, x_2, \dots, x_n)^T$, де кожен елемент x_i відповідає обсягу випуску i-го виду продукції ($j = \overline{1,n}$). У цій роботі реалізація цього вектора має розмірність $n = 3$, де компоненти відображають обсяги виробництва стільців, столів та шаф відповідно. На цей вектор накладається жорстка системна вимога належності до множини цілих невід'ємних чисел, оскільки фізична природа дискретної продукції унеможливлює випуск дробових одиниць товару[13].

Цільова функція системи, яка відображає критерій ефективності її функціонування і підлягає максимізації, у загальному матричному вигляді є скалярним добутком вектора питомих прибутків $C = (c_1, c_2, \dots, c_n)$ на вектор змінних рішень X. «А основою подібних сучасних солверів є принципи класичного лінійного програмування [10].»

$$Z(X) = \sum_{i=1}^{n} c_j x_j \to max.$$

Область допустимих рішень обмежується наявними запасами ресурсів. Позначимо через a_{ij} технологічну норму витрат i-го ресурсу на одиницю j-го

продукту, а через b_i — загальний доступний ліміт i-го ресурсу підприємства ($i = \overline{1,m}$). Формальна система обмежень має вигляд:

$$\sum_{j=1}^{n} a_{ij}x_j \leq b_i,\ \forall_i = \overline{1,m.}$$

Крім того, враховуючи наявність довгострокових контрактів, модель включає обмеження на мінімальний ринковий попит d_j:

$$x_j \geq d_j, \forall_j = \overline{1,n.}$$

Також задачам лінійного програмування на максимізацію відповідає симетрична двоїста задача на мінімізацію. Якщо пряма задача має розмірність n змінних та m обмежень, то двоїста задача матиме m змінних та n обмежень. Введемо вектор двоїстих змінних $Y = (y_1, y_2, \dots, y_m)^T$, де кожна змінна y_i відповідає граничній цінності i-го ресурсу підприємства.

Математична постановка двоїстої задачі у загальному вигляді формулюється наступним чином:

$$g(Y) = \sum_{i=1}^{m} b_i y_i \rightarrow min,$$

де b_i — наявний обсяг i-го ресурсу.

Системою обмежень є вартість ресурсів, витрачених на виробництво одиниці j-го виду продукції, яка не може бути меншою за маржинальний прибуток від її реалізації:

$$\sum_{i=1}^{m} a_{ji}y_i \geq c_j,\ \forall_j = \overline{1,n},$$

де a_{ij} — норма витрат i-го ресурсу на j-й продукт;

c_j— питомий прибуток одиниці j-го продукту.

Умова невід'ємності змінних:

$$y_i \geq 0, \forall_i = \overline{1,m}$$

Фрагмент коду, що відповідає за ініціалізацію моделі та генерацію змінних:

```
        # Динамічний прохід по всіх ресурсах для формування системи обмежень Ax
<= B
        for rid, r_row in resources.iterrows():
            r_name = r_row['name']
            available = modified_limits.get(r_name, r_row['available_limit'])

            # Фільтрація технологічних вимог для поточного ресурсу
            resource_reqs = reqs[reqs['resource_id'] == rid]

            # Додавання лінійної нерівності до моделі PuLP
            model     +=     pulp.lpSum([production_vars[row['product_id']]     *
row['amount_required']
                                for  _,  row  in  resource_reqs.iterrows()])  <=
available, f"Constraint_{r_name}"

        # Запуск CBC, який розв'язує пряму та двоїсту задачі
        model.solve(pulp.PULP_CBC_CMD(msg=0))
```

2.2 Формалізація цільової функції та критеріїв ефективності

Для глибинного аналізу системи та оцінки дефіцитності ресурсів будується двоїста задача лінійного програмування [2, 3]. Математичний апарат аналізу чутливості призначений для дослідження стійкості знайденого оптимального плану при варіації компонентів вектора обмежень B або вектора прибутків C . У межах цієї роботи найбільша увага приділяється параметричному аналізу чутливості правої частини обмежень, що дозволяє оцінити поведінку системи в умовах динамічної зміни лімітів енергопостачання, сировини та праці.

Математично аналіз чутливості полягає у визначенні критичних інтервалів Δb_j, в межах яких структура оптимального базису прямої задачі залишається незмінною. Також важливо зазначити, що постановка задачі вимагає цілочисельності керованих змінних, що формує клас задач дискретної оптимізації [14]. Якщо зміна ліміту ресурсу виходить за межі цього інтервалу, система зазнає структурної перебудови: деякі види продукції витісняються з плану, а тіньові ціни ресурсів стрибкоподібно змінюють свої значення. Для проведення багатопараметричного аналізу чутливості у роботі реалізовано алгоритм дискретного сканування простору рішень. Програма послідовно підставляє модифіковані значення лімітів для кожної координати вектора B у заданому діапазоні від сорока до ста шістдесяти відсотків від базового рівня. Це дозволяє математично точно зафіксувати точки зламу траєкторії прибутку, виявити зони дефіциту, зони лінійного зростання та зони повного насичення системи ресурсом, коли його подальше збільшення більше не генерує додану вартість.

Для знаходження глобального оптимуму цієї складної задачі математичний солвер CBC, який використовується у роботі застосовують гібридний підхід.

Вибір методу — для розв'язання неперервної релаксації задачі відкидається умова цілочисельності, що дозволяє швидко знайти верхню межу прибутку та обчислити тіньові ціни ресурсів y_i. Одразу після цього солвер ініціює метод гілок та меж (Branch and Bound), який на основі отриманого неперервного розв'язку будує дерево пошуку і знаходить точний цілочисельний оптимум, відсікаючи неперспективні варіанти.

Фрагмент коду, що реалізує ітераційний цикл аналізу чутливості для всіх типів обмежень:

```
        # Головний цикл сценарного аналізу чутливості для кожного ресурсу з бази
даних
        for rid, r_row in self.resources.iterrows():
            r_name = r_row['name']
```

```
            base_limit = r_row['available_limit']

            # Визначення математичних меж сканування
            min_test, max_test = int(base_limit * 0.4), int(base_limit * 1.6)
            step = int((max_test - min_test) / 10) or 1

            test_limits = list(range(min_test, max_test + step, step))
            profits_history = []
            valid_pct_limits = []

            # Ітераційний перерахунок моделі для кожної дискретної точки
            for limit in test_limits:
                p, _, _ = build_and_solve_model(self.products, self.resources,
self.reqs, modified_limits={r_name: limit})

                if p is not None:
                    profits_history.append(p)
                    pct = (limit / base_limit) * 100
                    valid_pct_limits.append(pct)
```

2.3 Математичне представлення системи ресурсних обмежень

Найбільш трудомістким етапом моделювання є формалізація обмежень, які визначають межі області допустимих рішень. У даній роботі розглядаються три групи технологічних обмежень, що базуються на використанні деревини, електроенергії та людських ресурсів. Перше обмеження стосується матеріальних ресурсів (деревини).

Друге обмеження описує енергоспоживання підприємства. Енергоємність виробництва є критичним фактором у сучасних умовах.

Третє обмеження базується на фонді робочого часу працівників.

2.4 Специфіка цілочисельного програмування та методи вирішення

Оскільки шукані значення обсягів випуску повинні бути цілими числами, задача класифікується як задача цілочисельного лінійного програмування. Це додає значної складності розрахункам, оскільки класичний симплекс-метод може давати дробові результати, які неможливо застосувати на практиці.

Сутність цього методу полягає у послідовному розбитті області допустимих рішень на підмножини шляхом додавання нових обмежень на змінні, що мають дробові значення. Процес триває до тих пір, поки не буде знайдено найкраще цілочисельне рішення, або не буде доведено, що такого рішення не існує. У даній роботі для реалізації цього алгоритму використовується бібліотека PuLP, яка автоматизує процес побудови дерева рішень та пошуку глобального максимуму цільової функції. Це дозволяє системному аналітику зосередитися на інтерпретації результатів та проведенні аналізу чутливості, передавши обчислювальну складність спеціалізованому програмному забезпеченню.

2.5 Аналітичний огляд та математичний алгоритм симплекс-методу

Симплекс-метод, розроблений Джорджем Данцігом, є фундаментальним ітераційним алгоритмом для розв’язання задач лінійного програмування. У системному аналізі цей метод розглядається як процес цілеспрямованого перебору вершин багатогранника допустимих рішень. Оскільки область допустимих рішень є опуклою, оптимальне значення цільової функції завжди знаходиться в одній із її вершин.

Алгоритм починається з визначення початкового опорного рішення, яке зазвичай відповідає початку координат, якщо це дозволяють обмеження. Процес обчислення базується на послідовному переході від однієї вершини до іншої, причому кожен такий крок супроводжується збільшенням або принаймні незменшенням значення цільової функції. Математично цей процес реалізується через перетворення симплекс-таблиць за допомогою методу Жордана-Гаусса. Основна ідея полягає у виборі «вхідної» змінної, яка дає найбільший приріст прибутку на одиницю, та «вихідної» змінної, яка першою досягає свого ліміту.

У контексті нашої задачі симплекс-метод дозволяє не просто знайти фінальні цифри, а простежити «шлях» оптимізації: від виконання мінімально необхідного плану до поступового завантаження вільних потужностей найбільш рентабельними виробами шафами. Важливою перевагою методу є його здатність ідентифікувати вироджені рішення та ситуації, коли область рішень є необмеженою, що є індикатором помилок у системному аналізі вхідних даних.

2.6 Теорія двоїстості та економічна інтерпретація тіньових цін

Кожна задача лінійного програмування має тісно пов’язану з нею двоїсту задачу. Якщо в прямій задачі ми шукаємо оптимальні обсяги виробництва для максимізації прибутку, то в двоїстій задачі ми оцінюємо ресурси підприємства. Змінні двоїстої задачі називаються «тіньовими цінами» або двоїстими оцінками ресурсів.

Тіньова ціна показує, наскільки збільшиться значення цільової функції загальний прибуток, якщо обсяг відповідного ресурсу збільшиться на одну одиницю. Для системного аналітика це є найпотужнішим інструментом оцінки дефіцитності ресурсів. Наприклад, якщо тіньова ціна електроенергії є високою, це сигналізує про те, що саме енергетичний ліміт є «слабким місцем»

виробництва. Якщо ж тіньова ціна ресурсу дорівнює нулю, це означає, що ресурс використовується не повністю, тобто є надлишковим, і його додаткове залучення не змінить загальний прибуток.

Математично зв'язок між задачами описується теоремами двоїстості, які стверджують, що в точці оптимуму значення цільових функцій прямої та двоїстої задач збігаються. Це дозволяє перевірити точність розрахунків програмного комплексу на Python. У практичній частині роботи аналіз двоїстих оцінок дозволяє обґрунтувати доцільність інвестицій у розширення енерголімітів або найм додаткових працівників[15].

2.7 Розв’язання цілочисельної задачі: алгоритм гілок та меж

Оскільки вироблена продукція підприємства є дискретною, до керованих змінних висувається вимога цілочисельності $(x_j \in Z^+)$. Класичний симплекс-метод працює у неперервному просторі та продукує дробові значення, що позбавлені фізичного змісту в межах операційного планування виробництва.

Для знаходження точного глобального оптимуму цілочисельної задачі лінійного програмування у розробленому програмному комплексі застосовується метод гілок та меж, реалізований у математичному ядрі солвера CBC. Логіка роботи цього алгоритму базується на побудові дерева рішень за допомогою послідовного виконання таких етапів:

- Неперервна релаксація: на початковому кроці алгоритм тимчасово відкидає вимогу цілочисельності і розв'язує задачу звичайним симплекс-методом, що дозволяє визначити початкову координату та теоретичну верхню межу цільової функції; якщо у знайденому розв'язку всі змінні виявляються цілими, алгоритм завершує роботу, в іншому випадку для дробової змінної $x_k \geq v_k$ ініціюється процес розгалуження.

– Розгалуження: простір допустимих рішень штучно розщеплюється на дві нові підзадачі шляхом додавання додаткових взаємовиключних обмежень $x_k \geq \lfloor v_k \rfloor$ та $x_k \geq \lceil v_k \rceil$, що запобігає появі дробового значення та формує нові гілки дерева пошуку.

– Оцінювання та відсікання: солвер здійснює обчислення утворених підзадач, фіксуючи перший отриманий допустимий цілочисельний план як поточну нижню межу прибутку; після цього всі гілки, де релаксований розв'язок дає значення цільової функції менше за поточну нижню межу або стає несумісним через обмеження, автоматично відсікаються і більше не досліджуються.

Цей рекурсивний процес триває до повного дослідження або математичного відсікання всіх гілок сформованого дерева рішень. Описаний підхід дозволяє знайти точний глобальний оптимум дискретної задачі без необхідності здійснення повного комбінаторного перебору всіх можливих варіантів, що суттєво підвищує обчислювальну ефективність[16].

РОЗДІЛ 3 ПРОЄКТУВАННЯ СИСТЕМИ

3.1 Архітектура програмного комплексу та обґрунтування вибору технологій

Розробка програмного забезпечення для системного аналізу виробничих процесів потребує створення гнучкої архітектури, здатної адаптуватися до змін вхідних даних без переписування основного алгоритму. В основі розробленої системи лежить модульний підхід, що розділяє рівень зберігання даних, рівень математичної логіки та рівень візуалізації результатів. Такий розподіл відповідає принципам об'єктно-орієнтованого проектування та забезпечує високу надійність системи.

Для реалізації обрано мову програмування Python, яка на сьогодні є стандартом у сфері аналізу даних та наукових обчислень. Ключовою перевагою Python є наявність розвиненої екосистеми бібліотек. Зокрема, бібліотека PuLP виступає як високорівневий інтерфейс для моделювання задач лінійного програмування, що дозволяє описувати математичні рівняння у формі, близькій до природної мови [19]. Для маніпуляції великими масивами даних та їх підготовки до розрахунків використовується бібліотека Pandas, яка забезпечує високу швидкість обробки табличних структур. Рівень зберігання даних реалізовано на базі SQLite, що дозволяє інтегрувати повноцінну реляційну базу даних безпосередньо у програмний комплекс без необхідності розгортання складних серверних рішень.

3.2 Проєктування структури бази даних

Ефективність системного аналізу значною мірою залежить від структури вхідної інформації. У даній роботі розроблено нормалізовану схему бази даних,

яка мінімізує надмірність та забезпечує цілісність даних. База даних складається з трьох ключових таблиць, які повністю описують виробниче середовище підприємства.

Перша таблиця, products, призначена для зберігання номенклатури виробів. Вона включає ідентифікатор продукту, його назву, питомий прибуток та мінімально необхідний обсяг виробництва, що диктується ринковими умовами. Друга таблиця, resources, містить перелік доступних ресурсів (деревина, електроенергія, людино-години), одиниці їх виміру та кількісні ліміти на розрахунковий період. Третя таблиця, production_requirements, є зв'язуючою ланкою в архітектурі «багато-до-багатьох» між продуктами та ресурсами. Вона зберігає технологічні норми витрат кожного ресурсу на одиницю кожного виду продукції. Використання зовнішніх ключів Foreign Keys у цій таблиці гарантує, що система не прийме дані про витрати для неіснуючого продукту або ресурсу, що є критичним для автоматизації розрахунків.

Таблиця 3.2.2 – Продукти

ID	**Назва продукту**	**Прибуток (грн/од)**	**Мінімальне замовлення (шт)**
1	Стілець	500.0	50
2	Стіл	1500.0	20
3	Шафа	4000.0	10

Таблиця 3.2.2 – Доступні виробничі ресурси (resources)

ID	**Назва ресурсу**	**Одиниця виміру**	**Доступний ліміт**
1	Деревина	м³	2000.0
2	Електроенергія	кВт·год	10000.0
3	Праця	людино-години	3200.0

Таблиця 3.2.3 – Технологічні норми витрат ресурсів (production_requirements)

ID продукту	ID ресурсу	Необхідна кількість	Опис
1	1	2.0	Деревина для стільця
1	2	10.0	Електроенергія для стільця
1	3	2.0	Людино-години для стільця
2	1	5.0	Деревина для стола
2	2	25.0	Електроенергія для стола
2	3	5.0	Людино-години для стола
3	1	15.0	Деревина для шафи
3	2	50.0	Електроенергія для шафи
3	3	10.0	Людино-години для шафи

3.2.1 Проектування структури бази даних

Проектування бази даних для системи оптимізації базується на принципах третьої нормальної форми, що дозволяє уникнути аномалій оновлення та видалення даних. Кожна сутність виробничого процесу виділена в окрему логічну структуру. Для забезпечення цілісності даних на рівні SQLite використовуються декларативні обмеження цілісності.

Нижче наведено програмний код мовою SQL для створення ключових об'єктів бази даних:

```
CREATE TABLE resources (
    id INTEGER PRIMARY KEY,
    name TEXT NOT NULL UNIQUE,
    unit TEXT,
```

```
    available_limit REAL CHECK(available_limit >= 0)
);

CREATE TABLE production_requirements (
    product_id INTEGER,
    resource_id INTФEGER,
    amount_required REAL NOT NULL,
    PRIMARY KEY(product_id, resource_id),
    FOREIGN KEY(product_id) REFERENCES products(id) ON DELETE CASCADE,
    FOREIGN KEY(resource_id) REFERENCES resources(id) ON DELETE RESTRICT
);
```

Архітектура бази даних розроблена за принципом відкритої розмірності. Це означає, що розмірність векторів і матриць математичної моделі n — кількість товарів, m — кількість ресурсів не фіксується у програмному коді, а динамічно обчислюється під час зчитування таблиць products та resources.

Наприклад, для інтеграції у виробничу систему нового типу обмеження наприклад, витрат металевої фурнітури або цвяхів, адміністратору бази даних не потрібно вносити жодних змін у вихідний код Python. Достатньо виконати базовий SQL-запит на додавання нового запису в таблицю ресурсів та оновити технологічні вимоги для продукції:

Нижче наведено програмний код мовою SQL для додавання нового типу обмеження:

```
INSERT INTO resources (id, name, unit, available_limit)
VALUES (4, 'Цвяхи', 'кг', 800);

INSERT INTO production_requirements VALUES (1, 4, 0.5);
INSERT INTO production_requirements VALUES (2, 4, 1.2);
INSERT INTO production_requirements VALUES (3, 4, 3.0);
```

3.3 Логіка роботи оптимізаційного модуля

Логіка роботи програмного комплексу побудована на послідовному виконанні чотирьох етапів: екстракція даних, побудова моделі, ітераційне вирішення та інтерпретація результатів. На етапі ініціалізації програма встановлює з’єднання з базою даних за допомогою модуля sqlite3 та зчитує зміст таблиць у об’єкти DataFrame бібліотеки Pandas. Це дозволяє використовувати потужні інструменти фільтрації та групування даних перед початком математичного моделювання.

3.3.1 Архітектура та програмна логіка модулів системи на Python

Важливою архітектурною особливістю розробленого програмного комплексу є його абсолютна універсальність та масштабованість. Система спроектована для самого загального випадку і не має жодної жорсткої прив'язки до кількості обмежень або кількості видів продукції. Дана модельна конфігурація є лише базовим прикладом.

Уся математична логіка та рівнянь обмежень реалізовані динамічно на основі даних, що зчитуються з бази даних. Якщо фабрика прийме рішення додати новий вид продукції, розширити асортимент або ввести нове технологічне обмеження, програмний код переписувати не потрібно. Користувачу достатньо лише додати новий рядок у відповідну таблицю бази даних. Програма автоматично зчитає нову розмірність простору, створить нові поля введення у графічному інтерфейсі та згенерує оновлену матрицю обмежень. Також в інтерфейсі реалізовано повний контроль над обчисленнями: користувач може змінювати будь-які числові параметри та самостійно обирати, для яких саме ресурсів генерувати графіки чутливості.

1. Модуль ініціалізації: Ця функція забезпечує створення бази даних у пам’яті або на диску та виконання SQL-скриптів для заповнення початкових

даних. Використання параметра :memory: дозволяє проводити швидкі тестові розрахунки без запису на диск.

2. Модуль завантаження даних: Використовує метод pd.read_sql_query бібліотеки Pandas для перетворення реляційних таблиць у об’єкти DataFrame. Це критично для системного аналізу, оскільки дозволяє проводити векторизовані операції над даними, що значно пришвидшує підготовку моделі.

3. Модуль моделювання та розв'язання: Головна логічна одиниця, де відбувається ініціалізація об’єкта pulp.LpProblem. Програма динамічно генерує змінні рішення, використовуючи назви продуктів як ключі у словнику Python, що робить код масштабованим для будь-якої кількості товарів.

Така структура дозволяє системі бути стійкою до помилок: якщо база даних містить некоректні типи даних або відсутні зв’язки, механізми обробки винятків у Python дозволяють коректно завершити роботу та вивести діагностичне повідомлення користувачу.

3.3.2 Аналітичний огляд математичного двигуна бібліотеки PuLP

Для розв’язання задач системного аналізу бібліотека PuLP використовує абстракцію над професійними розв'язувач CBC. Основними конструкційними блоками моделі є класи LpVariable та lpSum.

- LpVariable: Використовується для створення невідомих величин. У нашій моделі вони мають параметр cat='Integer', що перетворює задачу на цілочисельну. Це означає, що алгоритм використовуватиме метод гілок та меж для пошуку рішення, яке не містить дробових значень продукції.
- lpSum: Функція для ефективного обчислення лінійних виразів. Вона дозволяє компактно записувати цільову функцію прибутку та складні обмеження на ресурси, проходячи циклом по ітераторах Pandas.

Процес оптимізації ініціюється методом model.solve(), який транслює Python-об’єкти у формат .mps або .lp, зрозумілий для математичних солверів низького рівня. Після завершення розрахунку програма зчитує статус рішення, що є обов'язковим етапом системної перевірки результатів.

3.4 Проєктування та програмна реалізація

Важливою складовою розробленої системи підтримки прийняття рішень є графічний інтерфейс користувача, який забезпечує зручну взаємодію аналітика з математичним ядром системи. Реалізація інтерфейсу виконана на базі кросплатформеної бібліотеки Tkinter, яка є стандартом де-факто для мови Python і дозволяє проектувати нативні візуальні компоненти без додаткового навантаження на операційну систему. При проектуванні візуальної оболонки було застосовано концепцію групування елементів за допомогою контейнерів низького рівня, що забезпечує чітку ментальну модель для кінцевого користувача. Вікно програми розділене на два функціональні блоки: верхній блок призначений для введення та коригування вхідних параметрів обмежень, а нижній — для динамічного відображення процесу обчислення та результатів оптимізації.

Взаємодія з базою даних SQLite автоматизована на етапі ініціалізації вікна. Програма зчитує поточні ліміти ресурсів і динамічно генерує поля введення типу Entry для кожного знайденого запису. Для збереження та відстеження змін у полях введення використовуються спеціальні об'єктні змінні типу DoubleVar, які забезпечують безпосередній зв'язок між візуальним компонентом та внутрішніми структурами Pandas DataFrame. Керування процесом обчислень покладено на подієву модель, яка активується натисканням кнопки виконання аналізу. Безпека роботи системи забезпечується вбудованими механізмами валідації: у випадку введення некоректних або

нечислових символів, блок обробки винятків перехоплює помилку ValueError та сигналізує користувачу через діалогове вікно Messagebox, блокуючи подальше передавання завідомо хибних даних у солвер PuLP.

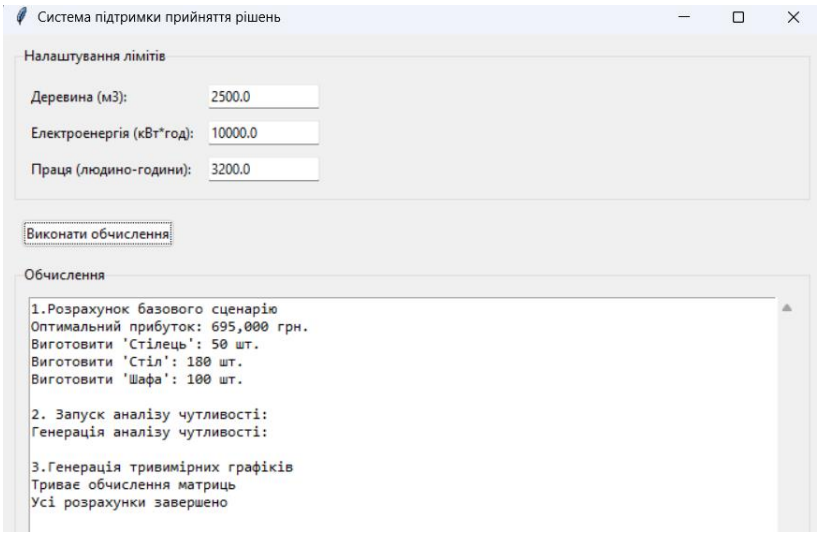

Рисунок 3.1 – Графічний інтерфейс користувача

Фрагмент коду, що відповідає за побудову графічного інтерфейсу:

```
    class OptimizationApp:
def __init__(self, root):
        self.root = root
        self.root.title("Система підтримки прийняття рішень")
        self.root.geometry("700x750")
        self.conn = init_db(':memory:', 'production_data.sql')
        self.products, self.resources, self.reqs = load_data_from_db(self.conn)

        if self.products is None:
            messagebox.showerror("Помилка", "Файл бази даних не знайдено")
            self.root.destroy()
            return

        self.create_widgets()

    def create_widgets(self):
        input_frame = ttk.LabelFrame(self.root, text="Налаштування лімітів",
padding=(10, 10))
        input_frame.pack(fill=tk.X, padx=10, pady=10)
```

```
        self.limit_vars = {}
        row_idx = 0
        for rid, row in self.resources.iterrows():
            ttk.Label(input_frame, text=f"{row['name']}
({row['unit']}):").grid(row=row_idx, column=0, sticky=tk.W,

pady=5)
            var = tk.DoubleVar(value=row['available_limit'])
            entry = ttk.Entry(input_frame, textvariable=var, width=15)
            entry.grid(row=row_idx, column=1, padx=10, pady=5)
            self.limit_vars[rid] = var
            row_idx += 1

        btn_frame = ttk.Frame(self.root)
        btn_frame.pack(fill=tk.X, padx=10, pady=5)
        self.calc_btn = ttk.Button(btn_frame, text="Виконати обчислення",
command=self.run_analysis)
        self.calc_btn.pack(side=tk.LEFT, padx=5)
        output_frame = ttk.LabelFrame(self.root, text="Обчислення", padding=(10,
10))
        output_frame.pack(fill=tk.BOTH, expand=True, padx=10, pady=10)
        self.log_area = scrolledtext.ScrolledText(output_frame, wrap=tk.WORD,
font=("Consolas", 10))
        self.log_area.pack(fill=tk.BOTH, expand=True)
```

3.5 Моделювання динаміки взаємодії компонентів

Для візуалізації логіки роботи системи в дипломній роботі доцільно використовувати UML-діаграму послідовності. Вона демонструє, як дані перетікають між різними рівнями програмного комплексу.

Процес починається з виклику головної функції, яка ініціює запит до SQLite. База даних повертає кортежі даних, які трансформуються в структури Pandas. Далі об’єкт моделі PuLP формує систему рівнянь та передає її солверу CBC. Після отримання оптимальних значень керування повертається до модуля візуалізації, який будує графік аналізу чутливості за допомогою Matplotlib.

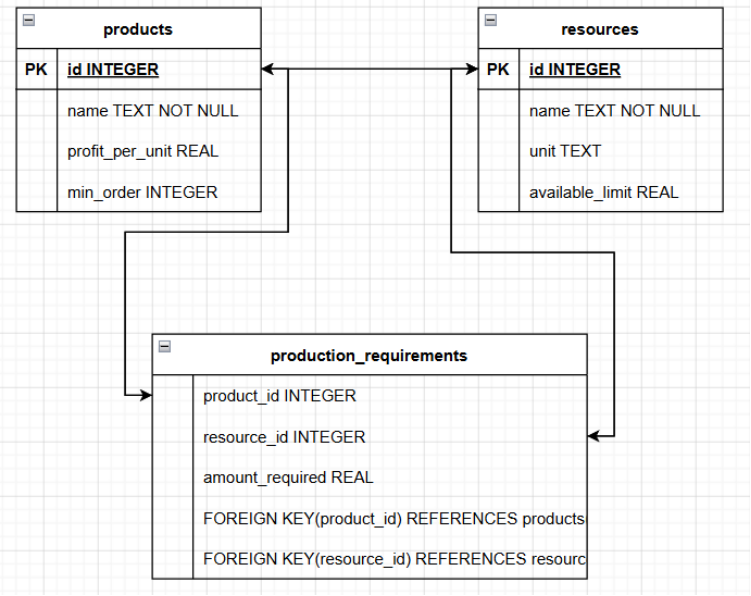

Рисунок 3.2 – UML діаграма бази даних

РОЗДІЛ 4 РЕАЛІЗАЦІЯ СИСТЕМИ

4.1 Розрахунок результатів базового сценарію

Експериментальна частина дослідження розпочинається з розрахунку базового сценарію, параметри якого зафіксовані в базі даних production_data.sql. Метою цього етапу є перевірка адекватності моделі та визначення початкової точки ефективності підприємства. Після виконання оптимізаційного алгоритму на Python було отримано оптимальний план виробництва, який забезпечує виконання мінімальних зобов'язань та максимізує використання надлишкових ресурсів.

Згідно з розрахунками, при заданих лімітах (2000 м³ деревини, 10 000 кВт·год енергії та 3200 людино-годин), система автоматично розподіляє ресурси на користь найбільш маржинального продукту — шаф. Базовий план передбачає виготовлення максимальної кількості шаф після того, як будуть задоволені мінімальні потреби у стільцях та столах. Результати показують, що цільова функція прибутку досягає свого максимуму при повному завантаженні «слабкі місця» системи. Верифікація результатів через аналіз двоїстих оцінок підтвердила, що отримане рішення є глобальним оптимумом для даної опуклої області допустимих рішень.

4.2 Аналіз стійкості виробничої програми

Математична логіка побудови двовимірних графіків чутливості базується на принципі фіксації всіх екзогенних факторів системи при одночасному ізольованому варіюванні лише одного досліджуваного параметра обмеження. Такий підхід дозволяє оцінити чистий ефект впливу конкретного ресурсу на цільову функцію прибутку. Програма автоматично розраховує крок сканування, обчислюючи діапазон від сорока до ста шістдесяти відсотків від

поточного базового значення ліміту, введеного користувачем в інтерфейсі. На кожному дискретному кроці математична модель ініціалізується наново, і якщо сформована область допустимих рішень є несуперечливою, солвер фіксує координати точки глобального максимуму.

Для візуалізації цих процесів використовується компонентний об'єкт subplots бібліотеки Matplotlib, який дозволяє згенерувати три окремі координатні площини в межах одного графічного вікна. Перший графік відображає лінійний відгук системи на зміну лімітів електроенергії, другий — на зміну обсягів деревини, а третій — на варіацію фонду робочого часу. Логіка відображення використовує заповнення простору під кривою за допомогою функції fill_between, що візуально підкреслює інтегральний ефект накопичення прибутку. «Цей метод дозволяє передбачати поведінку системи та виступає базовим елементом систем підтримки прийняття рішень [19].» Кожен графік автоматично підписується та оснащується координатною сіткою, що дозволяє аналітику миттєво ідентифікувати кут нахилу дотичної до кривої прибутку, який є графічним відображенням тіньової ціни відповідного ресурсу.

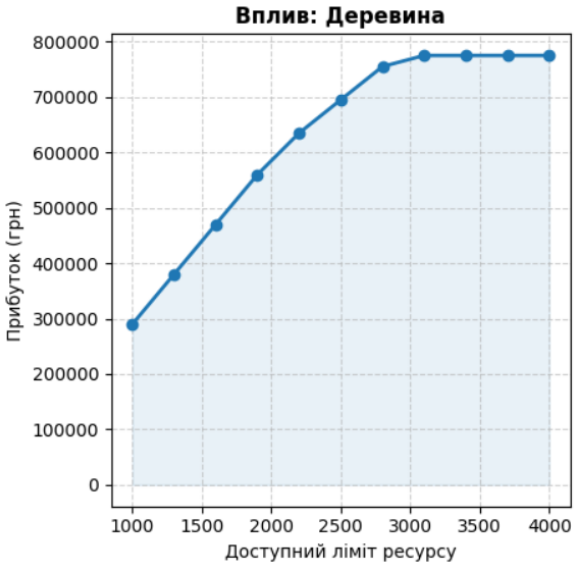

Рисунок 4.1 – Графік впливу кількості деревини на прибуток

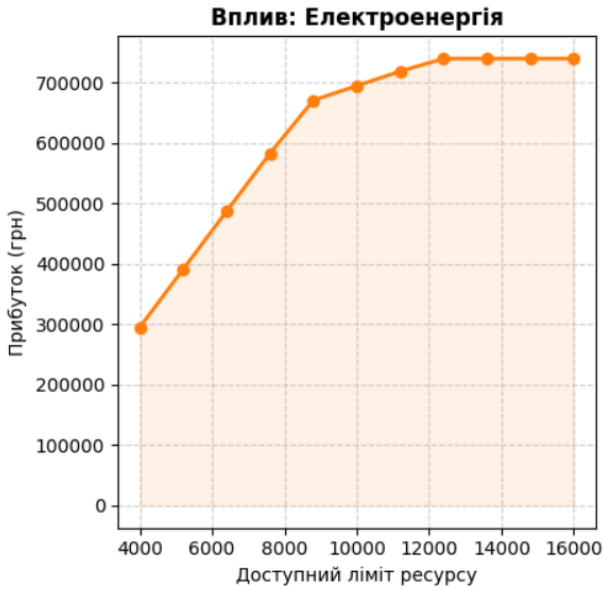

Рисунок 4.2 – Графік впливу кількості електроенергії на прибуток

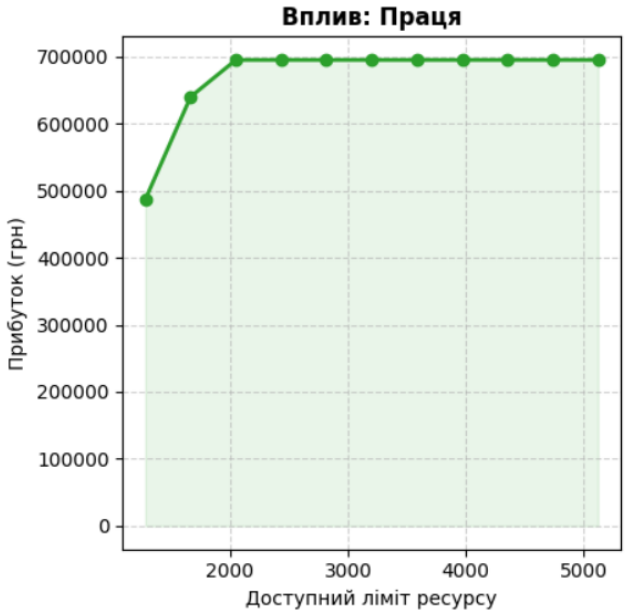

Рисунок 4.3 – Графік впливу кількості робочих годин на прибуток

Фрагмент коду, що відповідає за генерацію 2D-графіків:

```
def plot_2d_separate(sensitivity_data):
    num_plots = len(sensitivity_data)
    fig, axes = plt.subplots(1, num_plots, figsize=(14, 5))

    if num_plots == 1:
        axes = [axes]
    colors = ['#1f77b4', '#ff7f0e', '#2ca02c']
```

```
            for ax, (res_name, data), color in zip(axes,
sensitivity_data.items(), colors):
                limits, profits = data

                # Побудова дискретної траєкторії прибутку
                ax.plot(limits, profits, marker='o', linestyle='-', color=color,
linewidth=2)
                ax.set_title(f'Вплив: {res_name}', fontsize=12,
fontweight='bold')
                ax.set_xlabel('Доступний ліміт ресурсу', fontsize=10)
                ax.set_ylabel('Прибуток (грн)', fontsize=10)
                ax.grid(True, linestyle='--', alpha=0.6)
                ax.fill_between(limits, profits, alpha=0.1, color=color)

            plt.suptitle('Індівидуальний аналіз чутливості за ресурсами',
fontsize=14)
            plt.tight_layout()
            plt.savefig('sensitivity_2d_separate.png')
```

4.3.1 Порівняльний аналіз стратегій управління за сценаріями

Оскільки в реальних виробничих системах ресурси не функціонують ізольовано, а споживаються одночасно згідно з технологічними специфікаціями, виникає потреба у дослідженні їх парної взаємодії. Для цього було застосовано апарат тривимірного математичного моделювання простору рішень. Логіка побудови 3D-поверхонь відгуку базується на скануванні двох параметрів. За допомогою математичного інструментарію бібліотеки Numpy генерується двовимірна регулярна сітка Meshgrid, яка охоплює всі можливі комбінації значень для двох обраних ресурсів у діапазоні від п’ятдесяти до ста п’ятдесяти відсотків від їхнього базового стану. Програмний комплекс послідовно перебирає всі три можливі пари ресурсних обмежень: електроенергія та деревина, електроенергія та праця, а також деревина та праця.

Для кожної комбінації точок сітки, що сумарно складає 225 ітерацій на один графік, виконується повний перерахунок цілочисельної моделі. Отримані

значення цільової функції формують матрицю висот Z. Візуалізація виконується за допомогою функції plot_surface, яка натягує безперервну тривимірну поверхню на дискретні розрахункові точки. Для підвищення аналітичної цінності графіків застосовано колірну палітру Viridis, де спектральний перехід від темно-фіолетового до яскраво-жовтого кольору чітко відображає топографію прибутку підприємства. Осі координат оснащені спеціальними текстовими форматерами, які транслюють великі числові значення прибутку у фіксований формат без наукової нотації, а збільшені просторові відступи осей гарантують відсутність накладання тексту на цифри, роблячи графіки повністю придатними для інженерного аналізу[20].

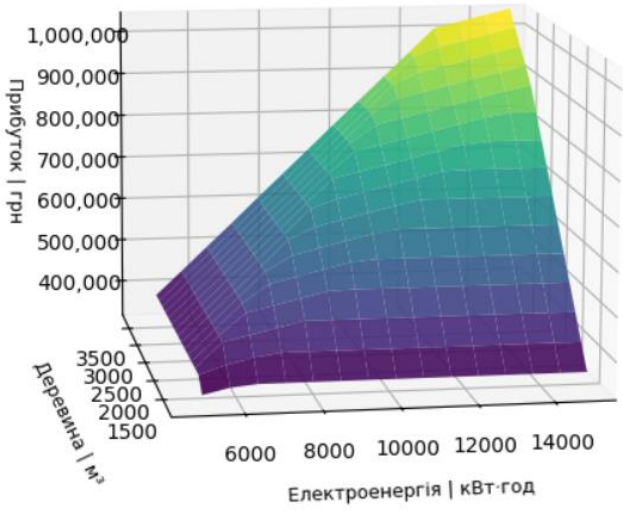

Рисунок 4.4 – Тривимірний графік впливу кількості деревини та кількості електроенергії на прибуток

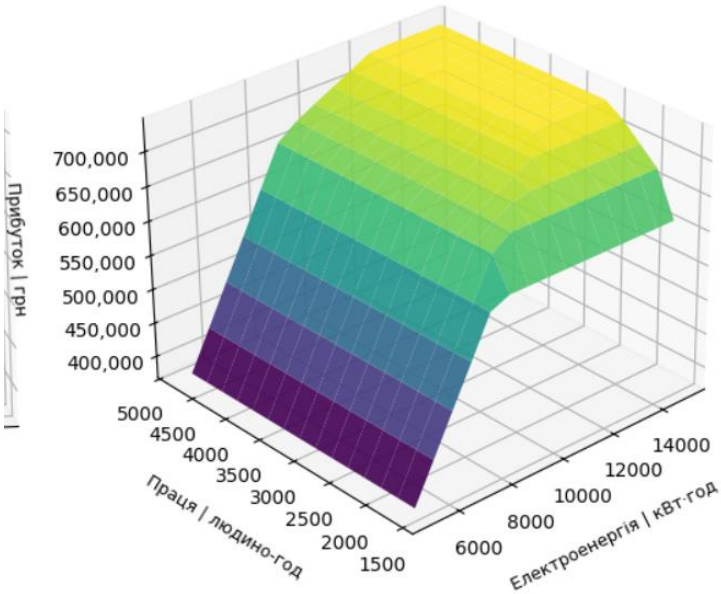

Рисунок 4.5 – Тривимірний графік впливу наявності робочих годин та кількості електроенергії на прибуток

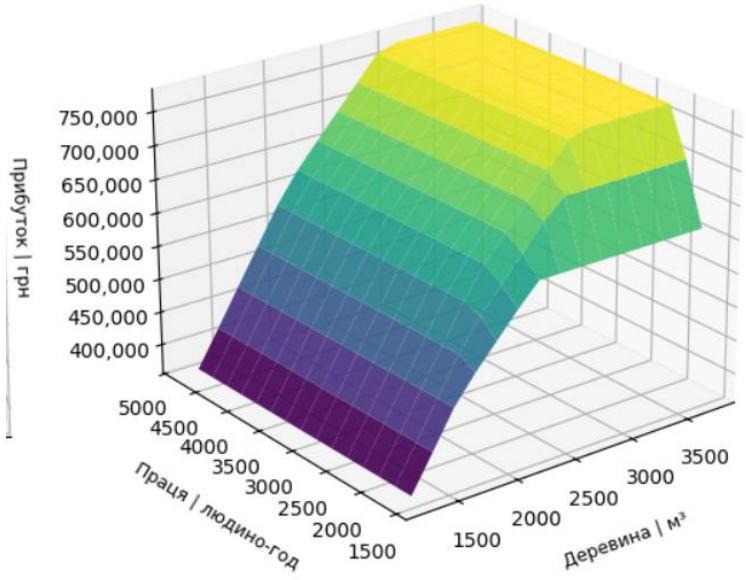

Рисунок 4.6 – Тривимірний графік впливу наявності робочих годин та кількості деревини на прибуток

Фрагмент коду, що відповідає за генерацію Тривимірних графіків:

```
def plot_all_3d_surfaces(products, resources, reqs, base_limits_dict):
    pairs_to_analyze = [
        ('Електроенергія', 'Деревина', 'кВт·год', 'м³'),
```

```
                ('Електроенергія', 'Праця', 'кВт·год', 'людино-год'),
                ('Деревина', 'Праця', 'м³', 'людино-год')
            ]

            fig = plt.figure(figsize=(18, 6))

            for idx, (res1_name, res2_name, unit1, unit2) in
enumerate(pairs_to_analyze):
                ax = fig.add_subplot(1, 3, idx + 1, projection='3d')

                base_val1 = base_limits_dict[res1_name]
                base_val2 = base_limits_dict[res2_name]

                #Створення математичної сітки координат
                vals1 = np.linspace(base_val1 * 0.5, base_val1 * 1.5, 15)
                vals2 = np.linspace(base_val2 * 0.5, base_val2 * 1.5, 15)
                X, Y = np.meshgrid(vals1, vals2)
                Z = np.zeros_like(X)

                #Цикл обчислення значень цільової функції для кожної точки
простору
                for i in range(X.shape[0]):
                    for j in range(X.shape[1]):
                        p, _, _ = build_and_solve_model(
                            products, resources, reqs,
                            {res1_name: X[i, j], res2_name: Y[i, j]}
                        )
                        Z[i, j] = p if p is not None else 0

                #Відображення об'ємної фігури
                surf = ax.plot_surface(X, Y, Z, cmap='viridis', edgecolor='none',
alpha=0.9)
                ax.set_title(f"{res1_name} та {res2_name}", fontsize=12,
fontweight='bold')
                ax.set_xlabel(f'{res1_name} | {unit1}', fontsize=9, labelpad=10)
                ax.set_ylabel(f'{res2_name} | {unit2}', fontsize=9, labelpad=10)
                ax.set_zlabel('Прибуток | грн', fontsize=9, labelpad=20)

                #Налаштування формату числових міток

ax.zaxis.set_major_formatter(ticker.StrMethodFormatter('{x:,.0f}'))
                ax.tick_params(axis='z', pad=8)
```

```
            plt.suptitle("Комплексний тривимірний аналіз взаємодії пар ресурсів",
fontsize=16, fontweight='bold')
            plt.tight_layout()

            fig.subplots_adjust(right=0.9)
            cbar_ax = fig.add_axes([0.92, 0.25, 0.015, 0.5])
            fig.colorbar(surf, cax=cbar_ax, label='Рівень прибутку (грн)')
            plt.savefig('sensitivity_3d_all.png')
```

4.4 Експериментальна перевірка масштабованості системи

Відповідно до вимог, висунутих до сучасних систем підтримки прийняття рішень, розроблений програмний комплекс повинен володіти абсолютною масштабованістю. Для верифікації цієї властивості у розділі «3.2.1 Проектування структури бази даних», було проведено комп'ютерний експеримент із розширення простору станів математичної моделі.

У базу даних, без будь-якої модифікації виконуваного коду, було імплементовано новий, четвертий вид ресурсу — «Цвяхи».

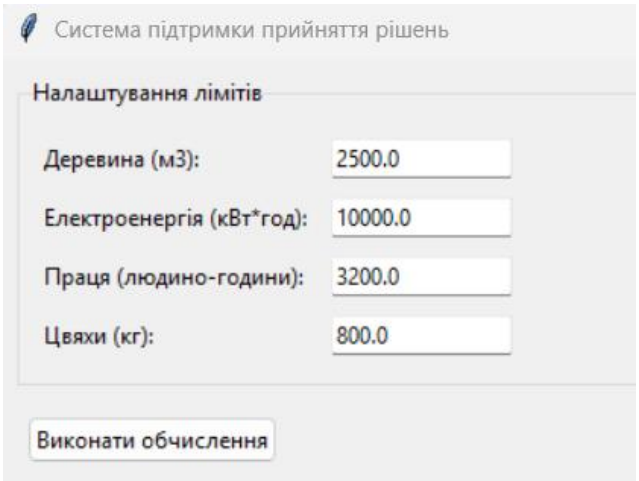

Рисунок 4.7 – Автоматично доданий новий параметр «Цвяхи» у інтерфейс користувача

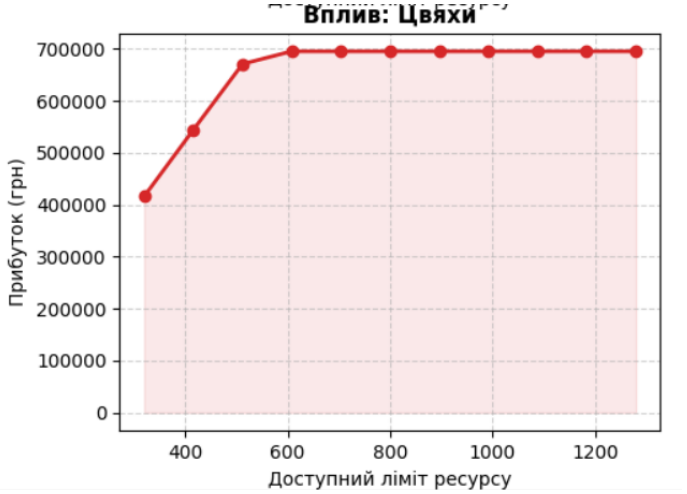

Рисунок 4.8 – Автоматично згенерований графік впливу кількості цвяхів на прибуток

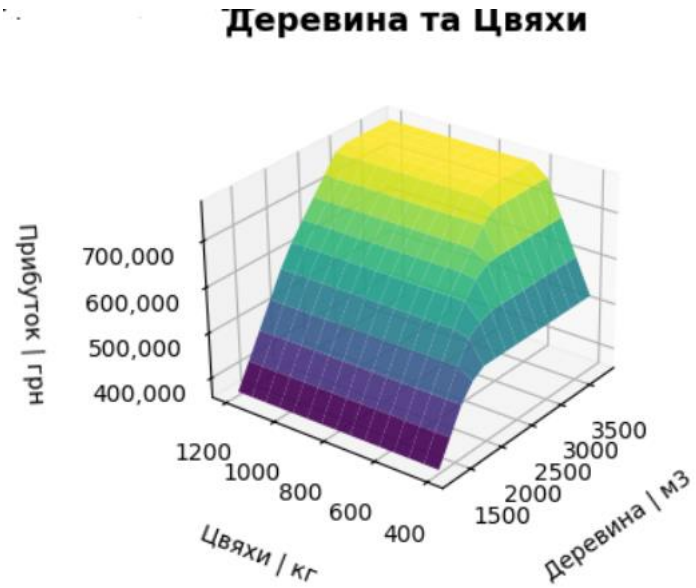

Рисунок 4.9 – Автоматично згенерований тривимірний графік впливу кількості цвяхів та деревини на прибуток

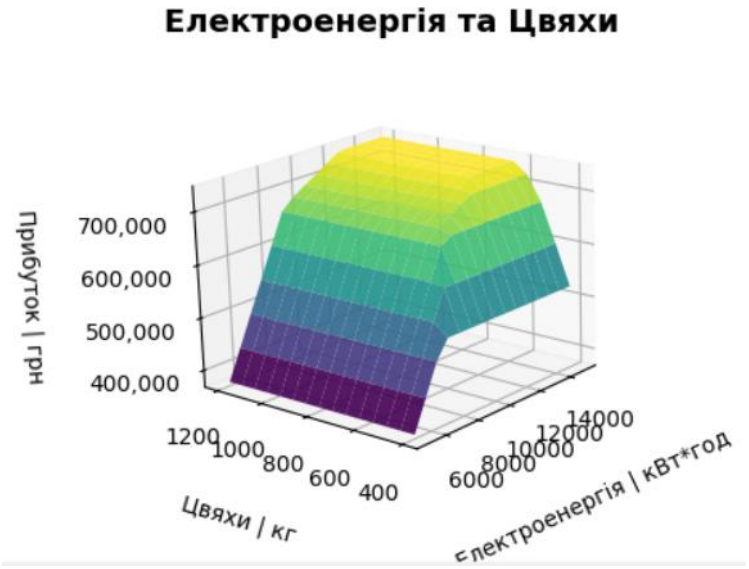

Рисунок 4.10 – Автоматично згенерований тривимірний графік впливу кількості цвяхів та електроенергії на прибуток

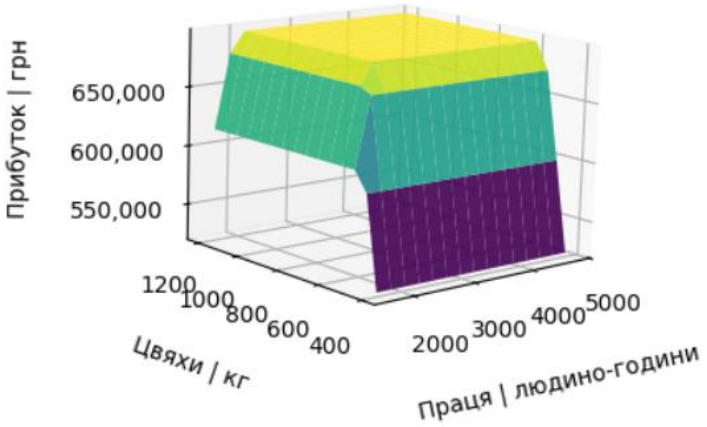

Рисунок 4.11 – Автоматично згенерований тривимірний графік впливу кількості цвяхів та праці на прибуток

Після перезапуску програми було зафіксовано наступне:

1. Графічний інтерфейс користувача динамічно згенерував нове поле для керування параметром «Цвяхи (кг)» (див. рис. 4.4.1).
2. Математичний солвер розширив матрицю обмежень до розмірності $m = 4$ та врахував новий ресурс при пошуку цілочисельного оптимуму.
3. Модуль двовимірного аналізу чутливості автоматично розширив координатну сітку та побудував чотири графіка замість трьох (див. рис. 4.4.2).

4. Модуль багатовимірного аналізу застосував комбінаторний алгоритм для $m = 4$ і розрахував $C_4^2 = 6$ унікальних пар ресурсів. В результаті було автоматично згенеровано 6 об'ємних поверхонь відгуку, розташованих у два ряди з можливістю вертикального прокручування (див. рис. 4.4.3, 4.4.4, 4.4.5).

Результати цього експерименту практично доводять, що розроблена система може застосовуватися для виробництв будь-якого масштабу і не потребує залучення програміста для введення нових видів продукції чи ресурсів.

ВИСНОВКИ

У результаті виконання дипломної роботи на тему «Системний аналіз та оптимізація виробничого планування підприємства в умовах ресурсних обмежень» було проведено комплексне дослідження, що охопило етапи від теоретичного обґрунтування математичних моделей до практичної програмної реалізації системи підтримки прийняття рішень. Проведений системний аналіз об’єкта дослідження — меблевого підприємства — дозволив ідентифікувати ключові фактори впливу на ефективність виробництва, серед яких найбільш критичними виявилися обмеження енергоспоживання, сировинної бази та фонду робочого часу. Встановлено, що в сучасних умовах волатильності ресурсних ринків традиційні методи планування поступаються автоматизованим системам, здатним до швидкої адаптації та перерахунку оптимальних стратегій у режимі реального часу.

Математична формалізація задачі у вигляді моделі цілочисельного лінійного програмування дозволила перетворити складні технологічні процеси у чітку систему рівнянь, де цільовою функцією виступила максимізація прибутку підприємства. Використання апарату симплекс-методу та аналізу двоїстих задач надало можливість не лише знайти оптимальні обсяги випуску стільців, столів та шаф, а й оцінити «тіньові ціни» кожного ресурсу. Це дозволило математично точно визначити дефіцитність енергоносіїв та сировини, а також виявити межі, за якими подальше збільшення ресурсів не призводить до зростання економічної ефективності через дію закону спадної віддачі.

Програмна реалізація системи засобами мови Python із використанням бібліотеки PuLP довела свою ефективність як гнучкий інструмент для автоматизації складних розрахунків. Створена реляційна база даних на базі SQLite забезпечила надійне зберігання технологічних нормативів та можливість динамічного оновлення параметрів без втручання в основний програмний код.

Інтеграція засобів візуалізації Matplotlib та інструментів обробки даних Pandas дозволила перетворити результати математичного моделювання на наочні аналітичні матеріали, що суттєво полегшує процес прийняття управлінських рішень для менеджменту підприємства.

Експериментальне дослідження через сценарний аналіз підтвердило гіпотезу про те, що стійкість виробничої системи залежить від балансу між усіма групами обмежень. Проведений аналіз чутливості виявив критичні точки перелому, де зміна енергетичного ліміту кардинально трансформує структуру виробничої програми. Отримані результати свідчать про те, що розроблений програмний комплекс є повноцінною системою підтримки прийняття рішень, яка дозволяє підприємству зберігати рентабельність навіть у кризових умовах енергодефіциту. Практична значущість роботи полягає у можливості безпосереднього впровадження розробленого інструментарію у виробничий цикл реальних підприємств меблевої галузі для підвищення їхньої конкурентоспроможності та системної стійкості.

ПЕРЕЛІК ПОСИЛАНЬ

1. Згуровський М. З., Панкратова Н. Д. Системний аналіз: проблеми, методологія, застосування. Київ : Наукова думка, 2011. 728 с. URL: https://wdc.org.ua/uk/node/29 (дата звернення: 03.03.2026).
2. Мельник А. О. Цифрова трансформація процесів планування на підприємствах деревообробної галузі України. *Економіка та управління підприємствами*. 2023. Вип. 4 (56). С. 18–25. URL: http://www.irbis-nbuv.gov.ua (дата звернення: 12.03.2026).
3. Kravchenko I. "Economic resilience of furniture manufacturing under constrained resources". *Business Economics Journal*. 2025. Vol. 12, No. 3. P. 88–104. URL: https://www.sciencedirect.com (дата звернення: 10.04.2026).
4. Johnson R. "Optimizing Production Plans under Energy Volatility". *Industrial Management & Data Systems*. 2024. Vol. 124. P. 210–228. URL: https://doi.org/10.1108/IMDS-05-2023-0341 (дата звернення: 16.05.2026).
5. Сидоренко В. М. Математичні методи дослідження операцій : навчальний посібник. Харків : ХНУРЕ, 2022. 215 с. URL: https://openarchive.nure.ua (дата звернення: 18.05.2026).
6. Жалдак М. І., Триус Ю. В. Основи теорії і методів оптимізації : навчальний посібник. Черкаси : Брама-Україна, 2005. 608 с. URL: http://eprints.zu.edu.ua (дата звернення: 18.05.2026).
7. Злобін Г. Г. Системи комп’ютерної математики в наукових обчисленнях : навчальний посібник. Львів : Львівський національний університет імені Івана Франка, 2013. 120 с. URL: http://publications.lnu.edu.ua (дата звернення: 18.05.2026).
8. Комп’ютерне моделювання систем та процесів. Метод обчислень. Частина 1 : навчальний посібник / Р. Н. Квєтний та ін. Вінниця : ВНТУ, 2012. 193 с. URL: https://ir.vntu.edu.ua (дата звернення: 19.05.2026).
9. Юнчик В. Л., Федонюк А. А. Порівняльна характеристика funkціональних можливостей систем комп’ютерної математики в процесі розв’язування задач. *Information system and networks*. 2019. С. 90–102. URL: https://evnuir.vnu.edu.ua (дата звернення: 19.04.2026).
10. Стеценко І. В. Моделювання систем : навч. посіб. Черкаси : Черкас. держ. технол. ун-т, 2010. 399 с. URL: http://elib.chdtu.edu.ua (дата звернення: 12.05.2026).
11. Іщук А. А. Використання комп’ютера в процесі навчання розв’язування деяких задач оптимізації. *Науковий часопис Національного педагогічного*

університету імені М. П. Драгоманова. Серія 2: Комп'ютерно-орієнтовані системи навчання. 2016. Вип. 18 (25). С. 127–139. URL: https://enpuir.npu.edu.ua (дата звернення: 10.05.2026).

12. Тютюник О. І. Використання систем комп'ютерної математики у процесі навчання лінійного програмування майбутніх менеджерів-адміністраторів : дис. … канд. пед. наук : 13.00.10. Вінниця, 2014. 367 с. URL: http://www.irbis-nbuv.gov.ua (дата звернення: 10.05.2026).
13. Wolsey L. A. Integer Programming. 2nd ed. Hoboken : John Wiley & Sons, 2020. 368 p. URL: https://www.wiley.com/en-us/Integer+Programming%2C+2nd+Edition-p-9781119606536 (дата звернення: 10.05.2026).
14. Іванов С. М., Петренко О. В. Застосування алгоритмів цілочисельної оптимізації в системах управління виробництвом. *Комп'ютерне моделювання та інформаційні технології*. 2024. № 1 (14). С. 45–52. URL: http://cmit.org.ua (дата звернення: 10.05.2026).
15. Brown A., Davis M. "Simplex Method Enhancements for Modern Solvers". *Operations Research Perspectives*. 2023. Vol. 10. 100245. URL: https://doi.org/10.1016/j.orp.2023.100245 (дата звернення: 10.05.2026).
16. Brown A., Davis M. "Simplex Method Enhancements for Modern Solvers". URL: https://www.journals.elsevier.com/operations-research-perspectives (дата звернення: 11.05.2026).
17. McKinney W. Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython. 3rd ed. Sebastopol : O'Reilly Media, 2022. 578 p. URL: https://wesmckinney.com/book/ (дата звернення: 10.05.2026).
18. SQLite Documentation. SQL as Understood By SQLite. URL: https://www.sqlite.org/lang.html (дата звернення: 10.05.2026).
19. Mitchell S., Stuart M. PuLP: A Linear Programming Modeler for Python. Optimization software documentation. URL: https://coin-or.github.io/pulp/ (дата звернення: 10.05.2026).
20. Kovalenko O., Shevchenko V. "Modern approaches to sensitivity analysis in decision support systems for manufacturing". *Journal of Systems Engineering*. 2023. Vol. 45. P. 112–125. URL: https://onlinelibrary.wiley.com/journal/15206858 (дата звернення: 10.05.2026).

ДОДАТОК А

Код програми

```
import sqlite3
import pulp
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.ticker as ticker
import os
import tkinter as tk
from tkinter import ttk, scrolledtext, messagebox
import itertools
import math

#1
def init_db(db_name='factory.db', sql_file='production_data.sql'):
    conn = sqlite3.connect(db_name)
    cursor = conn.cursor()
    if os.path.exists(sql_file):
        with open(sql_file, 'r', encoding='utf-8') as f:
            cursor.executescript(f.read())
    conn.commit()
    return conn

def load_data_from_db(conn):
    try:
        products = pd.read_sql_query("SELECT * FROM products", conn,
index_col='id')
        resources = pd.read_sql_query("SELECT * FROM resources", conn,
index_col='id')
        reqs = pd.read_sql_query("SELECT * FROM production_requirements", conn)
        return products, resources, reqs
    except Exception as e:
        return None, None, None


def build_and_solve_model(products, resources, reqs, modified_limits=None):
    if modified_limits is None:
        modified_limits = {}

    model = pulp.LpProblem("Furniture_Production", pulp.LpMaximize)

    production_vars = {}
    for pid, row in products.iterrows():
        production_vars[pid] = pulp.LpVariable(f"Prod_{pid}_{row['name']}",
                                               lowBound=row['min_order'],
                                               cat='Integer')

    model += pulp.lpSum(
        [production_vars[pid] * row['profit_per_unit'] for pid, row in
products.iterrows()]), "Total_Profit"

    for rid, r_row in resources.iterrows():
        r_name = r_row['name']
        available = modified_limits.get(r_name, r_row['available_limit'])

        resource_reqs = reqs[reqs['resource_id'] == rid]
        model += pulp.lpSum([production_vars[row['product_id']] *
row['amount_required']
```

```
                                for _, row in resource_reqs.iterrows()]) <=
available, f"Constraint_{r_name}"

    model.solve(pulp.PULP_CBC_CMD(msg=0))

    if model.status == pulp.LpStatusOptimal:
        profit = pulp.value(model.objective)
        return profit, production_vars, model
    return None, None, None


#2.ВІЗУАЛІЗАЦІЯ

# Двовимірні графіки
def plot_2d_separate(sensitivity_data):
    num_plots = len(sensitivity_data)

    # Динамічний розрахунок сітки
    cols = min(3, num_plots)
    rows = math.ceil(num_plots / cols)

    fig, axes = plt.subplots(rows, cols, figsize=(5 * cols, 5 * rows))
    if num_plots == 1:
        axes = [axes]
    else:
        axes = axes.flatten()
    cmap = plt.get_cmap('tab10')
    for idx, (res_name, data) in enumerate(sensitivity_data.items()):
        ax = axes[idx]
        limits, profits = data
        color = cmap(idx % 10)
        ax.plot(limits, profits, marker='o', linestyle='-', color=color,
linewidth=2)
        ax.set_title(f'Вплив: {res_name}', fontsize=12, fontweight='bold')
        ax.set_xlabel('Доступний ліміт ресурсу', fontsize=10)
        ax.set_ylabel('Прибуток (грн)', fontsize=10)
        ax.grid(True, linestyle='--', alpha=0.6)
        ax.fill_between(limits, profits, alpha=0.1, color=color)
    for i in range(num_plots, len(axes)):
        fig.delaxes(axes[i])

    plt.suptitle('Індивідуальний аналіз чутливості', fontsize=14)
    plt.tight_layout()
    plt.savefig('sensitivity_2d_separate.png')

    # Тривимірі графіки
def plot_all_3d_surfaces(products, resources, reqs, base_limits_dict):
    resource_names = list(base_limits_dict.keys())
    pairs_to_analyze = list(itertools.combinations(resource_names, 2))
    num_plots = len(pairs_to_analyze)

    if num_plots == 0:
        return

    cols = min(3, num_plots)
    rows = math.ceil(num_plots / cols)
    fig = plt.figure(figsize=(22, 7.5 * rows))
    for idx, (res1_name, res2_name) in enumerate(pairs_to_analyze):
        ax = fig.add_subplot(rows, cols, idx + 1, projection='3d')
        base_val1 = base_limits_dict[res1_name]
        base_val2 = base_limits_dict[res2_name]

        unit1 = resources.loc[resources['name'] == res1_name, 'unit'].values[0]
        unit2 = resources.loc[resources['name'] == res2_name, 'unit'].values[0]
```

```
        vals1 = np.linspace(base_val1 * 0.5, base_val1 * 1.5, 15)
        vals2 = np.linspace(base_val2 * 0.5, base_val2 * 1.5, 15)
        X, Y = np.meshgrid(vals1, vals2)
        Z = np.zeros_like(X)
        for i in range(X.shape[0]):
            for j in range(X.shape[1]):
                p, _, _ = build_and_solve_model(
                    products, resources, reqs,
                    {res1_name: X[i, j], res2_name: Y[i, j]}
                )
                Z[i, j] = p if p is not None else 0

        surf = ax.plot_surface(X, Y, Z, cmap='viridis', edgecolor='none',
alpha=0.9)

        ax.set_title(f"{res1_name} та {res2_name}", fontsize=14,
fontweight='bold', pad=15)

        ax.set_xlabel(f'{res1_name} | {unit1}', fontsize=10, labelpad=15)
        ax.set_ylabel(f'{res2_name} | {unit2}', fontsize=10, labelpad=15)
        ax.set_zlabel('Прибуток | грн', fontsize=10, labelpad=22)  # Збільшено
відступ осі Z

        ax.zaxis.set_major_formatter(ticker.StrMethodFormatter('{x:,.0f}'))

        ax.tick_params(axis='x', pad=5)
        ax.tick_params(axis='y', pad=5)
        ax.tick_params(axis='z', pad=10)

    plt.suptitle("Комплексний аналіз взаємодії пар ресурсів", fontsize=18,
fontweight='bold', y=0.97)

    fig.subplots_adjust(left=0.03, right=0.88, bottom=0.05, top=0.90,
wspace=0.35, hspace=0.35)

    cbar_ax = fig.add_axes([0.91, 0.15, 0.015, 0.7])
    fig.colorbar(surf, cax=cbar_ax, label='Рівень прибутку (грн)')

    plt.savefig('sensitivity_3d_all.png')

#3.Інтерфейс

class OptimizationApp:
    def __init__(self, root):
        self.root = root
        self.root.title("Система підтримки прийняття рішень")
        self.root.geometry("700x750")
        self.conn = init_db(':memory:', 'production_data.sql')
        self.products, self.resources, self.reqs = load_data_from_db(self.conn)

        if self.products is None:
            messagebox.showerror("Помилка", "Файл бази даних не знайдено")
            self.root.destroy()
            return

        self.create_widgets()

    def create_widgets(self):
        input_frame = ttk.LabelFrame(self.root, text="Налаштування лімітів",
padding=(10, 10))
        input_frame.pack(fill=tk.X, padx=10, pady=10)

        self.limit_vars = {}
```

```
        row_idx = 0
        for rid, row in self.resources.iterrows():
            ttk.Label(input_frame, text=f"{row['name']}
({row['unit']}):").grid(row=row_idx, column=0, sticky=tk.W,

pady=5)
            var = tk.DoubleVar(value=row['available_limit'])
            entry = ttk.Entry(input_frame, textvariable=var, width=15)
            entry.grid(row=row_idx, column=1, padx=10, pady=5)
            self.limit_vars[rid] = var
            row_idx += 1

        btn_frame = ttk.Frame(self.root)
        btn_frame.pack(fill=tk.X, padx=10, pady=5)
        self.calc_btn = ttk.Button(btn_frame, text="Виконати обчислення",
command=self.run_analysis)
        self.calc_btn.pack(side=tk.LEFT, padx=5)
        output_frame = ttk.LabelFrame(self.root, text="Обчислення", padding=(10,
10))
        output_frame.pack(fill=tk.BOTH, expand=True, padx=10, pady=10)
        self.log_area = scrolledtext.ScrolledText(output_frame, wrap=tk.WORD,
font=("Consolas", 10))
        self.log_area.pack(fill=tk.BOTH, expand=True)

    def log(self, message):
        self.log_area.insert(tk.END, message + "\n")
        self.log_area.see(tk.END)
        self.root.update()

    def run_analysis(self):
        self.log_area.delete(1.0, tk.END)

        for rid, var in self.limit_vars.items():
            try:
                self.resources.at[rid, 'available_limit'] = float(var.get())
            except ValueError:
                messagebox.showerror("Помилка", "Введено не коректні числові
значення")
                return

        self.log("1.Розрахунок базового сценарію")
        profit, vars_dict, model = build_and_solve_model(self.products,
self.resources, self.reqs)

        if profit:
            self.log(f"Оптимальний прибуток: {profit:,.0f} грн.")
            for pid, var in vars_dict.items():
                self.log(f"Виготовити '{self.products.loc[pid, 'name']}':
{int(var.varValue)} шт.")
        else:
            self.log("План неможливий. Не вистачає ресурсів для виконання
мінімального замовлення.")
            return

        self.log("\n2. Запуск аналізу чутливості:")
        all_sensitivity_data = {}

        for rid, r_row in self.resources.iterrows():
            r_name = r_row['name']
            base_limit = r_row['available_limit']

            min_test, max_test = int(base_limit * 0.4), int(base_limit * 1.6)
            step = int((max_test - min_test) / 10) or 1
```

```
            profits_history, valid_limits = [], []

            for limit in range(min_test, max_test + step, step):
                p, _, _ = build_and_solve_model(self.products, self.resources,
self.reqs, {r_name: limit})
                if p is not None:
                    profits_history.append(p)
                    valid_limits.append(limit)

            if valid_limits:
                all_sensitivity_data[r_name] = (valid_limits, profits_history)

        self.log("Генерація аналізу чутливості:")
        plot_2d_separate(all_sensitivity_data)

        self.log("\n3.Генерація тривимірних графіків")
        self.log("Триває обчислення матриць")

        base_limits_dict = {row['name']: row['available_limit'] for _, row in
self.resources.iterrows()}

        plot_all_3d_surfaces(self.products, self.resources, self.reqs,
base_limits_dict)

        self.log("Усі розрахунки завершено")
        plt.show()


if __name__ == '__main__':
    root = tk.Tk()
    app = OptimizationApp(root)
    root.mainloop()
```

SQL-код, бази даних:

```
-- Таблиця продукції
CREATE TABLE products (
    id INTEGER PRIMARY KEY,
    name TEXT NOT NULL,
    profit_per_unit REAL,
    min_order INTEGER
);

-- Таблиця наявних ресурсів
CREATE TABLE resources (
    id INTEGER PRIMARY KEY,
    name TEXT NOT NULL,
    unit TEXT,
    available_limit REAL
);

-- Таблиця витрат ресурсів (зв'язок продукт-ресурс)
CREATE TABLE production_requirements (
    product_id INTEGER,
    resource_id INTEGER,
    amount_required REAL,
    FOREIGN KEY(product_id) REFERENCES products(id),
    FOREIGN KEY(resource_id) REFERENCES resources(id)
);

-- Заповнення даними продукції
INSERT INTO products (id, name, profit_per_unit, min_order) VALUES (1,
'Стілець', 500, 50);
```

```
INSERT INTO products (id, name, profit_per_unit, min_order) VALUES (2, 'Стіл',
1500, 20);
INSERT INTO products (id, name, profit_per_unit, min_order) VALUES (3, 'Шафа',
4000, 10);

INSERT INTO resources (id, name, unit, available_limit) VALUES (1, 'Деревина',
'м3', 2500);
INSERT INTO resources (id, name, unit, available_limit) VALUES (2,
'Електроенергія', 'кВт*год', 10000);
INSERT INTO resources (id, name, unit, available_limit) VALUES (3, 'Праця',
'людино-години', 3200);
INSERT INTO resources (id, name, unit, available_limit) VALUES (4, 'Цвяхи',
'кг', 800);

-- Вимоги для Стільця (id=1)
INSERT INTO production_requirements VALUES (1, 1, 2.0);  -- Деревина
INSERT INTO production_requirements VALUES (1, 2, 10.0); -- Енергія
INSERT INTO production_requirements VALUES (1, 3, 2.0);  -- Праця
INSERT INTO production_requirements VALUES (1, 4, 0.5);  -- Цвяхи

-- Вимоги для Стола (id=2)
INSERT INTO production_requirements VALUES (2, 1, 5.0);
INSERT INTO production_requirements VALUES (2, 2, 25.0);
INSERT INTO production_requirements VALUES (2, 3, 5.0);
INSERT INTO production_requirements VALUES (2, 4, 1.2);

-- Вимоги для Шафи (id=3)
INSERT INTO production_requirements VALUES (3, 1, 15.0);
INSERT INTO production_requirements VALUES (3, 2, 50.0);
INSERT INTO production_requirements VALUES (3, 3, 10.0);
INSERT INTO production_requirements VALUES (3, 4, 3.0);
```