ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ
НАВЧАЛЬНО–НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
Кафедра інженерії програмного забезпечення

Пояснювальна записка
до магістерської роботи
на ступінь вищої освіти магістр
на тему «**Удосконалення інформаційної технології пошуку оптимального рішення на основі методів оптимізації**»

Виконав: студент 6 курсу, групи ПДМ-61
спеціальності:
121 Інженерія програмного забезпечення
(шифр і назва спеціальності)

Колодюк А. В.
(прізвище та ініціали)

Керівник Жебка В. В.
(прізвище та ініціали)

Рецензент
(прізвище та ініціали)

Нормоконтроль Трінтіна Н.А.
(прізвище та ініціали)

КИЇВ — 2021

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ
НАВЧАЛЬНО–НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
Кафедра Інженерії програмного забезпечення
Ступінь вищої освіти - «Магістр»
Спеціальність - 121 Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри
інженерії програмного забезпечення
_____ Негоденко О.В.
“____” _____2021 року

ЗАВДАННЯ
НА МАГІСТЕРСЬКУ РОБОТУ СТУДЕНТУ

Колодюк Андрій Васильович
(прізвище, ім’я, по батькові)

1. Тема роботи: «Удосконалення інформаційної технології пошуку оптимального рішення на основі методів оптимізації»
Керівник роботи доктор технічних наук, доцент Жебка В. В.
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)
затверджені наказом закладу вищої освіти від “11” жовтня 2021 р.№170.
2. Строк подання студентом роботи_____
3. Вхідні дані до роботи:
Науково-технічна література щодо розв’язування задач лінійного програмування.
Середовище розробки Qt Creator, Qt Design Studio, Qt Linguist.
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити).
4.1 Аналіз існуючих рішень для розв’язування задач лінійного програмування
4.2 Розробка алгоритмів для розв’язування задач симплекс-методом та транспортної задачі .
4.3 Проектування графічного інтерфейсу для взаємодії користувача із програмою .
5. Перелік графічного матеріалу
1. Титульний слайд .
2. Мета, об’єкт дослідження, актуальність обраної тем .
3. Короткі теоретичні відомості .
4. Технічне завдання .
5. Діаграма взаємодії користувача з програмою .
6. Проектування графічного дизайну .
7. Приклад розв’язування задачі .

6. Дата видачі завдання 30.11.2021 .

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів магістерської роботи	Строк виконання	Примітка
1	Вибір теми магістерської роботи		
2	Розробка завдання на магістерську роботу		
3	Вибір оптимальних програмних рішень для розробки		
4	Складання календарного плану та узгодження з науковим керівником		
5	Збір теоретично матеріалу		
6	Написання реферату та вступу		
7	Участь у наукових конференціях		
8	Написання першого розділу та відправка на перевірку науковому керівнику		
9	Написання другого розділу та відправка на перевірку науковому керівнику		
10	Написання третього розділу, висновку, презентації диплому і відправка на перевірку науковому керівнику		

Студент _____ Колодюк А. В.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Жебка В. В.
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Текстова частина магістерської роботи 70 с., 19 рис., 15 джерел, 16 таблиць, 4 додатки.

Метою роботи є підвищення ефективності визначення оптимального рішення за допомогою інформаційної технології розробленої на основі методів оптимізації.

Для досягнення мети було сформульовано ти вирішено наступні завдання:

1. Ознайомлення із теоретичними відомостями щодо розв’язування задач симплекс-методом та транспортною задачею.
2. Аналіз існуючих рішень. Щоб розробити якісне програмне забезпечення необхідно ознайомитися із уже готовими програмами, розглянути їх переваги та недоліки.
3. Вибір середовища розробки та мови програмування.
4. Розробка алгоритмів для розв’язування поставлених задач.
5. Конструювання графічного дизайну для взаємодії користувача та програми.

Об’єкт дослідження: пошук оптимального рішення на основі методів оптимізації.

Предмет дослідження: інформаційна технологія розв'язування задач лінійного програмування.

КЛЮЧОВІ СЛОВА: СИМПЛЕКС-МЕТОД, ТРАНСПОРТНА ЗАДАЧА, ДОДАТОК, МЕТОД ЛАГРАНЖА, EXCEL, MAPLE, MANTLAB, MATHEMATICA, QT, WINDOWS.

ЗМІСТ

ВСТУП

Щодня ми вирішуємо одні і ті ж проблеми: як досягти найбільшого ефекту, володіючи при цьому обмеженими засобами, адже наші можливості і ресурси завжди обмежені.

Для досягнення максимального ефекту при обмежених коштах, необхідно спланувати свої дії так, щоб досягти найкращого результату, витративши при цьому найменше ресурсів.

В XX столітті було створено математичний апарат, який дозволяє зробити планування більш точно. Новий розділ математики називається математичним програмуванням, яке з програмуванням для ЕОМ схожий тільки тим, що більшість завдань математичного програмування дуже складні для ручного розрахунку, вирішити їх можна тільки за допомогою ЕОМ, попередньо написавши програму.

У зв'язку із швидким розвитком технологій попит на спеціалістів у сфері інформаційних розробок стрімко зростає, тому підготовка фахівців повинна бути не тільки швидкою, а й якісною. Дуже великий відсоток працівників зосереджені на максимізації прибутку компацій, у яких вони працюють. Для цього щодня необхідно вирішувати безліч задач, які б дозволили скоротити витрати ресурсів. Для цього необхідно володіти певними методами і теоріями.

Основною метою проекту "Дослідження операцій. Методи лінійного програмування" є розробка програмного забезпечення для більш якісного вивчення розділів лінійного програмування, таких як "Транспортна задача" та "Симплекс-метод". Кожен із цих розділів вимагає від користувача вирішення цілого комплексу задач для отримання правильного результату, а усі дії перевірятимуться комп'ютером та, у випадку помилок, буде запропоновано варіанти вирішення та підказки до поточного кроку. Такий підхід до вивчення елементів лінійного програмування дозволить ефективніше готувати спеціалістів, підтримуватиме та розвиватиме системне мислення, а також усуне вірогідність помилки під час навчання.

Об'єктом дослідження є розробка програмного забезпечення для вивчення ітераційного розв'язування задач лінійного програмування симплекс-методом та транспортною задачею.

Предметом дослідження є програмне забезпечення, яке дозволить користувачам навчитися розв'язувати задачі лінійного програмування на практиці.

Створення нової програми – досить трудомістке завдання, особливо зараз, коли звичайний об'єм програмного забезпечення перевищує тисячі операторів. Майбутні фахівці в області розробки програмних забезпечень повинні мати уявлення про методи аналізу, проектування, реалізації та тестування програмних систем, вони також мають орієнтуватися в підходах і технологіях, які існують.

Розрозроблювані алгоритми і методи повинні враховувати динамічні зміни у вихідних даних і вносити коригування в обчислювальний процес в ході пошуку рішення. Новий програмний комплекс повинен зберігати ефективну працездатність навіть тоді, коли вихідна система обмежень стає суперечливою (випадок невласних задач ЛП) або неповною (випадок задач ЛП з неформалізованими обмеженнями)

1 АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ

1.1 Основи лінійного програмування

Методи оптимізації – це дисципліна, що охоплює широке коло як класичних екстремальних завдань (без обмежень і з ними), так і часто алгоритми лінійного програмування, а також транспортні та мережеві моделі, методи прогнозування та прийняття рішень.

Кожен із нас час від часу стикається із проблемами вибору, тобтоситуаціями, коли досягнення результату можливе не єдиним шляхом. У такому разі природньо обрати найкращий спосіб здійснити цю дію. Але поняття найкращий розв’язок, у свою чергу, залежить від ситуаціїі критерія, за яким цей розв’язок обирається. Так, для людини, що направляється з пункту А у пункт Б, різні способи зробити це – пітипішки, викликати таксі чи скористатися суспільним транспортом – відповідно будуть найкращими, якщо ця людина прагне дістатися якнайдешевше, якнайшвидше чи обрати найкраще співвідношенняціна-якість, де під якістю розуміється час подорожі.

Математично, такі задачі представляються оптимізаційними математичними моделями, що включають аналітичну форму критерію, що називається цільовою функцією, та області пошукурозв’язку, і які називаються оптимізаційними задачами. У деяких випадках область пошуку містить невелику кількість допустимих розвв’язків, тому розв’язок задачі оптимізації можливо знайти звичайним перебором можливих варіантів рішень, але більш загальною є ситуація, коли допустима множина містить нескінченну, незлічену чи зліченну, кількість варіантів. У такому випадку задачу розв’язують наближено чи точно, застосовуючи певну, як правило ітераційну, схему (алгоритм, метод), що ґрунтується на активному використанні математичної моделі задачі. Найбільш розвинутою областю теорії оптимізації є лінійна оптимізація чи лінійне програмування (Linear Programming), де розглядаються оптимізаційні задачі із лінійними цільовими функціями і

областю пошуку, що задається лінійними обмеженнями, тобто є багатогранником чи багатогранною областю. Це пояснюєтьсятим, що лінійне програмування має надзвичайно широку сферупрактичного використання у теорії прийняття рішень, дослідженні операцій і оптимальному плануванні. Інтерес до методів лінійного програмування викликаний також ітим, що лінійні оптимізаційні задачі виникають як допоміжні у іншихобластях теорії оптимізації, таких, як нелінійне і дискретнепрограмування. Отже, і на сьогоднішній день інтерес до методівлінійного програмування не слабшає.

Ефективність роботи сучасних підприємств, які є складними системами, залежить від якості організаційного управління. Процеси прийняття рішень лежать в основі будь-якої цілеспрямованої діяльності. При формуванні стратегічних і тактичних рішень керівник повинен враховувати безліч часом суперечливих міркувань, спиратися на складні критерії ефективності шляхів досягнення кінцевих цілей. У зв'язку з цим виникла необхідність застосовувати для аналізу і синтезу економічних ситуацій і систем математичні методи і сучасну обчислювальну техніку. Такі методи об'єднуються під загальною назвою – математичне програмування. Завдання математичного програмування знаходять застосування в різних областях людської діяльності, де необхідний вибір одного з можливих варіантів дій, наприклад при вирішенні багаточисельних проблем управління і планування виробничих процесів, в задачах проектування та перспективного планування, приорганізації функціонування та розвитку соціальних процесів, їх координації згосподарськими та економічними процесами. Оптимальні (ефективні) рішеннядозволяють досягати мети при мінімальних витратах трудових, матеріальних ісировинних ресурсів. Найбільш розробленим в теперішній час розділом математичного програмування є лінійне програмування яке застосовується при розробці методів відшукання екстремуму (максимуму або мінімуму) лінійних функцій декількох змінних при лінійних обмеженнях, накладених на змінні. За типом вирішуваних завдань його методи можна розділити на універсальні і спеціальні. За

допомогою універсальних методів (наприклад, симплекс-метод) можуть вирішуватися будь-які завдання лінійного програмування. Спеціальні методи враховують особливості цільової функції і системи обмежень.

Методи і моделі лінійного програмування широко застосовуються приоптимізації процесів у всіх галузях народного господарства: при розробцівиробничої програми підприємства, розподілі її по виконавцях, при розміщеннізамовлень між виконавцями і по тимчасових інтервалах, при визначеннінайкращого асортименту продукції, що випускається, в задачах перспективного,поточного та оперативного планування і управління; при плануваннівантажопотоків, визначенні плану товарообігу і його розподіл; в задачах розвиткуі розміщення продуктивних сил, баз і складів матеріальних ресурсів. Наприклад, технологічний гігант Amazon впроваджує в свою роботу найсучасніші рішення. Так, нещодавно компанія представила свій новий склад восьмого покоління, в якому здійснюється повністю автоматизована доставка товару зі складів до сортувальників. Декілька сотень мініатюрних роботів пересуваються по складах і переміщують на собі невеликі (за складськими мірками) стелажі, на яких знаходяться необхідні товари. Самі співробітники складу при цьому нікуди не ходять: вони стоять і чекають, коли робот привезе їм необхідний стелаж. Роботи переміщаються строго по прямій, зчитуючи своє місце зі штрих-кодів на підлозі. Щоб повернути, роботи Kiva зупиняються і повертають свій корпус. Стелаж при цьому залишається в тому ж положенні. Іншим прикладом може слугувати українська компанія Fozzy Group, яка розпочала проект по роботизації складської зони у розподільних центрах. Роботи будуть займатися переміщенням товарів у межах складу для подальшого збереження, відбору і завантаження. Це дозволить компанії збільшити продуктивність обробки товарів і забезпечити високу точність їх збору. Мета роботизації - підвищення ефективності людського ресурсу, використання його для роботи у напрямках, де потрібно більш інтелектуальний і персональний підхід, відзначили у компанії.

Актуальніть розробки полягає у більш ефективній підготовці спеціалістів, використовуючи найсучасніші методи навчання – комп’ютери, візуалізацію даних та контроль за процесом навчання.

Для ефективного засвоєння основних теоретичних положень та кращого розуміння методів розв'язування прикладів і задач під час користування програмним забезпеченням надаються короткі відомості з теорії систем лінійних рівнянь та нерівностей, а також теорії про транспортну задачу, методи розподілу ресурсів та можливості оптимізації поточного плану.

В останні роки важлива роль при викладанні багатьох дисциплін відводиться використанню пакетів прикладних програм для персональних комп 'ютерів. Їх використання дозволяє зробити всі необхідні дії з дослідження проблеми, аналізу даних, моделювання, документування й оформлення результатів, а також їх перевірки. Це дає можливість нез аглиблюватися у складання складних математичних моделей, вибір методів, труднощі програмування, а зосередитись на вирішенні конкретної задачі.

Програмне забезпечення призначено для вивчення математичного апарату за допомогою комп'ютера. Використання комп'ютерів звільняє студентів від складних обчислень, дозволяє сконцентрувати свою увагу не тільки на алгоритмі обчислення, а й безпосередньо на аналізі результатів і їх інтерпретації. Очевидно, що ефективність вивчення предмета підвищується, якщо у студента є можливість самостійно перебирати декілька варіантів, змінювати параметри, шукати інші способи розв’язку. Зараз необов'язково володіти професійними навичками програміста для розв'язання задач певного типу або знати декілька мов програмування. Цілком досить підібрати вже готову програму, яка реалізує певний алгоритм. Хороше програмне забезпечення дозволяє зробити всі необхідні дії для дослідження проблеми, аналізу даних, моделювання, документування й оформлення результатів у одному місці.

Таким чином, розроблене програмне забезпечення дозволить користувачам ефективно освоїти методи лінійного програмування, дізнатися

теоретичні відомості, а також на практиці розв’язувати завдання, будуючи необхідні таблиці у графічному вікні, а також формулювати автоматичний звіт про розв’язану задачу із детальними коментарями до кожного кроку. Можливіть збереження оцінки дозволяє відправити результат викладачеві, що значно пришвидшує навчальний процес та економить багато часу, оскільки не потрібно перевіряти роботи вручну.

1.2 Основні методи вирішення задач ЛП

1.2.1 Симплекс-метод

Симплекс-метод - універсальний метод вирішення завдань ЛП: цілеспрямований перебір рішень, відповідних вершин багатогранника області допустимих рішень.

Алгоритм вирішення задачі симплекс-методом:

1. Приведення задачі до канонічного вигляду
2. Знаходження початкового опорного рішення з "одиничним базисом"
3. Обчислення оцінки розкладів векторів за базисом опорного рішення і заповнення симплекс-таблиці
4. При отриманні ознаки єдиності оптимального рішення розв’язок задачі закінчується
5. При виконанні умови існування безлічі оптимальних рішень методом перебору знаходять всі оптимальні рішення

Симплекс-метод відноситься до числа кінцевих і монотонних методів, а саме: через кінцеве число кроків або ми отримаємо оптимальний план, або переконаємося в необмеженості цільової функції на безлічі планів завдання, причому послідовність симплексних таблиць будується так, що значення цільової функції монотонно зростають (в завданню максимізації) або монотонно зменшується (в задачі мінімізації).

По-перше, метод передбачає, що відома крайня точка. (Якщо екстремальна точка не вказана, варіант симплексного методу, який називається фазою I, використовується, щоб знайти його або визначити, що немає можливих рішень.) Далі, використовуючи алгебраїчну специфікацію задачі, тест визначає, чи це крайня точка є оптимальною. Якщо тест на оптимальність не пройдено, то шукається сусідня екстремальна точка вздовж ребра в напрямку, для якого значення цільової функції зростає з найбільшою швидкістю. Іноді можна рухатися вздовж ребра і безмежно збільшувати значення цільової функції. Якщо це відбувається, процедура закінчується призначенням краю, по якому об’єктив прямує до позитивної нескінченності. Якщо ні, то досягається нова екстремальна точка, яка має принаймні таке ж високе значення цільової функції, як і попередня. Потім описана послідовність повторюється. Припинення відбувається, коли знайдена оптимальна екстремальна точка або виникає необмежений випадок. Хоча в принципі необхідні кроки можуть зростати експоненціально з кількістю крайніх точок, на практиці метод зазвичай сходить до оптимального рішення за кілька кроків, які є лише невеликими кратними кількості крайніх точок.

Щоб проілюструвати симплексний метод, розглянемо приклад фабрики, яка виробляє два вироби, x1 і x2. Якщо прибуток другого типу вдвічі перевищує прибуток першого, то x1 + 2x2 представляє загальний прибуток. Функція x1 + 2x2 відома як цільова функція.

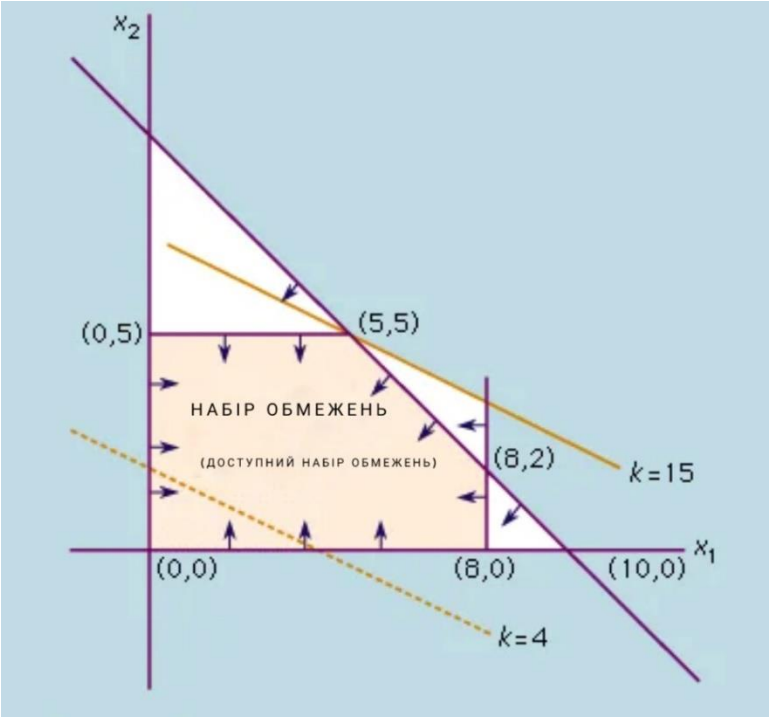


Рисунок 1.1 Застосування симплекс методу на практиці

Очевидно, що прибуток буде найвищим, якщо фабрика спрямує всі свої виробничі потужності на виготовлення другого виду товару. Однак у практичній ситуації це може бути неможливим; набір обмежень вноситься такими факторами, як наявність машинного часу, робочої сили та сировини. Наприклад, якщо для другого типу товару необхідна сировина, яка обмежена так, що в будь-якій партії можна виготовити не більше п’яти, то x2 має бути менше або дорівнювати п’яти; тобто, $x2 \leq 5$. Якщо перший товар вимагає іншого типу матеріалу, що обмежує його вісьмома на партію, то $x1 \leq 8$. Якщо x1 і x2 займають однаковий час для виготовлення і наявний машинний час дозволяє зробити максимум 10 у партії, то x1 + x2 має бути менше або дорівнювати 10; тобто $x1 + x2 \leq 10$.

Два інших обмеження полягають у тому, що кожне x1 і x2 має бути більшим або рівним нулю, оскільки неможливо створити від’ємне число з будь-якого; тобто $x1 \geq 0$ і $x2 \geq 0$. Завдання полягає в тому, щоб знайти значення x1 і x2, для яких прибуток є максимальним. Будь-яке рішення можна позначити парою чисел (x1, x2); наприклад, якщо x1 = 3 і x2 = 6, розв’язок (3,

6). Ці числа можуть бути представлені точками, нанесеними на двох осях, як показано на малюнку. На цьому графіку відстань вздовж горизонтальної осі представляє x1, а відстань уздовж вертикальної — x2. Через обмеження, наведені вище, можливі рішення повинні лежати в певній чітко визначеній області графіка. Наприклад, обмеження $x1 \geq 0$ означає, що точки, що представляють можливі рішення, лежать на або праворуч від осі x2. Аналогічно, обмеження $x2 \geq 0$ означає, що вони також лежать на або вище осі x1. Застосування всієї сукупності обмежень дає можливу множину розв'язків, яка обмежена многокутником, утвореним перетином прямих $x1 = 0$, $x2 = 0$, $x1 = 8$, $x2 = 5$ і $x1 + x2 = 10$. Для Наприклад, виробництво трьох позицій товару x1 і чотирьох товарів x2 є можливим рішенням, оскільки точка (3, 4) знаходиться в цій області. Однак, щоб знайти найкраще рішення, цільову функцію $x1 + 2x2 = k$ наносять на графік для деякого значення k, скажімо, $k = 4$. Це значення позначено ламаною лінією на малюнку. При збільшенні k створюється сімейство паралельних прямих, і лінія для $k = 15$ просто торкається обмеження, встановленого в точці (5, 5). Якщо k ще більше збільшити, то значення x1 і x2 будуть лежати за межами набору можливих рішень. Таким чином, найкращим рішенням є те, що виробляється однакова кількість кожного товару. Не випадково оптимальне рішення виникає у вершині, або «крайній точці» області. Це завжди буде справедливо для лінійних задач, хоча оптимальне рішення може не бути унікальним. Таким чином, розв'язання таких задач зводиться до пошуку того, яка крайня точка (або точки) дає найбільше значення цільової функції.

У симплекс-методі задача спочатку оформляється канонічним шляхом перетворення лінійних нерівностей у рівності шляхом введення «слабих змінних» $x3 \geq 0$ (так що $x1 + x3 = 8$), $x4 \geq 0$ (так що $x2 + x4 = 5$), $x5 \geq 0$ (так, щоб $x1 + x2 + x5 = 10$), і змінна x0 для значення цільової функції (так, щоб $x1 + 2x2 - x0 = 0$). Потім задачу можна переформулювати як знайти невід'ємні величини x1, …, x5 і найбільшу можливу x0, що задовольняє отримані рівняння. Одним з очевидних рішень є встановлення цільових змінних $x1 = x2$

= 0, що відповідає екстремальній точці в початку координат. Якщо одну з цільових змінних збільшити з нуля, а іншу зафіксувати на нульовому рівні, цільове значення x0 збільшиться за бажанням (за умови слабкості змінних, які задовольняють обмеженням рівності). Змінна x2 дає найбільше збільшення x0 на одиницю зміни; тому він використовується першим. Його збільшення обмежується вимогою невід'ємності до змінних. Зокрема, якщо x2 збільшити понад 5, x4 стане від'ємним.

При x2 = 5 ця ситуація створює нове рішення — (x0, x1, x2, x3, x4, x5) = (10, 0, 5, 8, 0, 5) — яке відповідає крайній точці (0, 5) на малюнку. Система рівнянь переводиться в еквівалентну форму шляхом розв'язування для ненульових змінних x0, x2, x3, x5 в термінах цих змінних, які тепер рівні нулю; тобто x1 і x4. Таким чином, нова цільова функція дорівнює x1 − 2x4 = −10, тоді як обмеження становлять x1 + x3 = 8, x2 + x4 = 5 і x1 − x4 + x5 = 5. Тепер очевидно, що збільшення x1 при утриманні x4, що дорівнює нулю, призведе до подальшого збільшення x0. Обмеження невід'ємності на x3 не дозволяє x1 вийти за межі 5. Нове рішення—(x0, x1, x2, x3, x4, x5) = (15, 5, 5, 3, 0, 0) — відповідає крайній точці (5, 5) на малюнку. Нарешті, оскільки розв'язування для x0 у термінах змінних x4 і x5 (які наразі мають нульове значення) дає x0 = 15 − x4 − x5, можна побачити, що будь-яка подальша зміна цих змінних зменшить цільове значення. Отже, оптимальне рішення існує в крайній точці (5, 5).

Критерій оптимальності стверджує:

Якщо цільовий рядок таблиці не має нульових записів у стовпцях, позначених основними змінними, і немає негативних записів у стовпцях, позначених неосновними змінними, тоді рішення, представлене таблицею, є оптимальним.

Як ми знаємо, що це гарантує, що рішення є оптимальним?

Ось основний принцип: число в стовпці і в рядку цілей повідомляє вам, як змінюється поточне значення цілі, якщо змінна і введе основу.

Давайте візьмемо приклад. Припустимо, що ви максимізуєте, і рядок цілей виглядає так

1 4 0 0 2 0 | 12

Це кодує цільове рівняння Z+4x1+0x2+0x3+2x4+0x5=12, або, як альтернатива, Z=12−4x1+0x2+0x3−2x4+0x5.

Тут основними змінними є x2,x3 та x5, а небазовими змінними є x1 та x4.

Тепер давайте розглянемо два випадки, враховуючи ключовий принцип, про який я згадував вище.

Будь-яка базова змінна вже міститься в базі, тому введення однієї з них в базу не змінює цільового значення. Ось чому в стовпцях, позначених базовими змінними, є нулі.

Якщо жодна з неосновних змінних не має негативних записів, то жодна з них не має позитивних коефіцієнтів, коли ви переписуєте цільове рівняння у другий спосіб. Тож дозволити будь-якому з них увійти в основу означає, що ви не будете збільшувати значення цільової функції. Наприклад, якщо дозволити x1 ввести базу, це призведе до зменшення Z на 4 для кожної одиниці, яка зменшується x1. Таким чином, якщо всі небазові змінні мають додатні або нульові записи, то ви не можете збільшити цільову функцію, дозволивши одній з них увійти в основу.

Отже, якщо цільовий рядок таблиці не має нульових записів у стовпцях, позначених основними змінними, і немає негативних записів у стовпцях, позначених неосновними змінними, то немає способу збільшити значення цільової функції, змінивши основу. . Оскільки зміна основи – це те, як ви змінюєте рішення, не повинно бути рішень, кращих за поточне. Отже, рішення, представлене таблицею, має бути оптимальним.

Недолік симплекс-методу: необхідність перерахунку всіх коефіцієнтів і вільних членів повної системи рівнянь обумовлена тим, що формули симплекс-методу передбачають обчислення всіх шуканих коефіцієнтів на n-

му кроці за даними k - vo кроку. Це збільшує обсяг розрахунків і уповільнює відшукання оптимального рішення.

Оптимальний план, знайдений звичайним симплекс-методом, як правило, не є цілочисельним. Розроблено алгоритми, набагато ефективніші, ніж звичайний симплекс-метод.

1.2.2 Метод множників Лагранжа

Розв'язуючи задачі оптимального управління, доводиться враховувати нелінійний характер взаємозв'язків між показниками. У загальному вигляді нелінійна економіко-математична модель має вигляд:

$$Z = f(x_1, x_2, \dots, x_n) \rightarrow \max(\min \quad), \tag{1.1}$$

за умов: $q_1(x_1, x_2, \dots, x_n)\{\leq = \geq\} b_1$, $(i = \overline{1, m})$, (1.2)

де $f(x_1, x_2, \dots, x_n)$ і $q_1(x_1, x_2, \dots, x_n)$ – нелінійні функції. (1.3)

Задачу нелінійного програмування намагаються звести до лінійного вигляду, проте тоді можливі значні похибки.

Для розв'язування задач нелінійного програмування не існує універсального методу, а тому доводиться застосовувати багато методів і обчислювальних алгоритмів, які ґрунтуються на теорії диференціального числення, і вибір їх залежить від конкретної постановки задачі та форми економіко-математичної моделі.

Методи нелінійного програмування бувають прямі та непрямі. Прямими методами оптимальні розв'язки відшукують у напрямку найшвидшого збільшення (зменшення) цільової функції. Типовими для цієї групи методів є градієнтні. Непрямі методи полягають у зведенні задачі до такої, знаходження оптимуму якої вдається спростити. До них належать, насамперед, найбільш розроблені методи квадратичного та сепарабельного програмування.

Оптимізацію з обмеженнями-рівностями виконують методами зведеного градієнта, наприклад методом Якобі, та множників Лагранжа.

Розглянемо метод множників Лагранжа на прикладі такої задачі нелінійного програмування:

$Z = f(x_1, x_2, \dots, x_n) \to \max(\min \quad)$, (1.4)

за умов: $q_1(x_1, x_2, \dots, x_n)\{\leq = \geq\} b_1$, $(i = \overline{1, m})$, (1.5)

де $f(x_1, x_2, \dots, x_n)$ і $q_1(x_1, x_2, \dots, x_n)$ – диференційовані. (1.6)

Ідея методу множників Лагранжа полягає в заміні даної задачі простішою: на знаходження екстремуму складнішої функції, але без обмежень. Ця функція називається функцією Лагранжа і подається у вигляді:

$L(x_1, x_2, \dots, x_n; \lambda_1, \lambda_2, \dots, \lambda_m) = f(x_1, x_2, \dots, x_n) + \sum_{i=1}^{m} \lambda_i (b_i - q_i(x_1, x_2, \dots, x_n))$, (1.7)

де λ_i — не визначені поки що величини, так звані множники Лагранжа.

Знайшовши частинні похідні функції L за всіма змінними і прирівнявши їх до нуля:

$\frac{\partial L}{\partial x_j} = 0$ $(j = \overline{1, n})$, *(1.8)*

$\frac{\partial L}{\partial \lambda_i} = 0 \quad (i = \overline{1, m})$, *(1.9)*

запишемо систему

$\frac{\partial f(x_1, x_2, \dots, x_n)}{\partial x_j} + \sum_{i=1}^{m} \lambda_i \frac{\partial q_i(x_1, x_2, \dots, x_n)}{\partial x_j} = 0 \qquad (j = \overline{1, n})$, *(1.10)*

$b_i - q_i(x_1, x_2, \dots, x_n) = 0 \qquad (i = \overline{1, m})$. *(1.11)*

що є, як правило, нелінійною.

Розв'язавши цю систему, знайдемо $X^* = (x_1, x_2, \dots, x_n)$ і $\lambda = (\lambda_1, \lambda_2, \dots, \lambda_m)$ — стаціонарні точки. Оскільки їх визначено з необхідної умови екстремуму, то в них можливий максимум або мінімум. Іноді стаціонарна точка є точкою перегину (сідлова точка). Отже, для визначення достатніх умов екстремуму та діагностування його типу існує спеціальний алгоритм

Звичайно метод має певні переваги та недоліки.

Переваги:

— метод найпростіший в розумінні і організації обчислювального процесу;

— має найбільшу точність інтерполяції;

— використання многочленів невисокого порядку і внаслідок цього мале число похибок в процесі обчислень.

Недоліки:

— при збільшенні числа вузлів і ступеня інтерполяційний многочлен Лагранжа потрібно будувати заново;

— повільна швидкість збіжності;

— найбільш складний серед методів інтерполяції в організації обчислювального процесу.

1.2.3 Транспортна задача

Одна з найбільш поширених завдань математичного програмування - транспортна задача, яка часто використовується для вирішення проблем управління витратами в транспортній сфері.

У транспортній задачі необхідно скласти план перевезень, що найбільш економно забезпечує задоволення попиту всіх пунктів споживання за рахунок реалізації всього продукту, виробленого пунктами виробництва. Замість пунктів виробництва і споживання відповідно можуть розглядатися пункти відправлення і призначення.

На сьогодні відомо кілька алгоритмів вирішення транспортної задачі лінійного програмування. Умовно їх можна розбити на 2 групи:

— перша основана на принципі послідовного поліпшення плану, коли обраний початковий план за допомогою розрахунків покращується до тих пір, поки він не стане оптимальним. Одним з методів першої групи є метод потенціалів;

— друга основана на методі послідовного скорочення нев'язок. Ця група основана на методі дозволених доданків.

Недоліком транспортної задачі ТЗ є те, що модель перевезень не враховує неоднорідність вантажів і транспортних засобів. Це можна врахувати шляхом розгляду багатоіндексних транспортних завдань. Реалізація даних завдань забезпечується рішенням трипланарної (Т-3Р) і триаксіальной (Т-3А)

транспортних задач. Їх рішення дозволяє отримати оптимальний план транспортування різних типів вантажів різними типами автотранспортних засобів.

Транспортна задача має наступний вигляд:

$$\begin{cases} \sum_{j=1}^{m} x_{ij} \leq a_i \\ \sum_{i=1}^{n} x_{ij} \leq b_j \\ x_{ij} \geq 0, \quad i = 1, n \end{cases} \tag{1.12}$$

Наприклад:

Розглянемо три компанії (Company1, Company2 і Company3), які виробляють мобільні телефони та розташовані в різних регіонах.

Аналогічно розглянемо три міста (а саме CityA, CityB і CityC), куди перевозяться мобільні телефони.

Компанії, де доступні мобільні телефони, відомі як джерела, а міста, куди перевозяться мобільні телефони, називаються пунктами призначення.

Припустимо,

- Компанія 1 виробляє одиниці a1,
- Компанія 2 виробляє одиниці a2,
- Компанія 3 виробляє одиниці a3,
- попит у місті A становить b1 одиниць,
- попит у місті B становить b2 одиниць,
- попит у місті C становить b3 одиниць.

Вартість транспортування від кожного джерела до пункту призначення наведена в таблиці:

Таблиця 1.1 Практичне вирішення ТЗ

	Місто A	Місто B	Місто C	Пропозиція
Компанія 1	C_{1A}	C_{1B}	C_{1C}	a_1
Компанія 2	C_{2A}	C_{2B}	C_{2C}	a_2
Компанія 3	C_{3A}	C_{3B}	C_{3C}	a_3
Попит	b_1	b_2	b_3	$\Sigma a_i = \Sigma b_i$

Перевезення мобільних телефонів повинно здійснюватися таким чином, щоб загальна вартість транспортування була мінімальною.

Існує два типи транспортних проблем:

і) Проблема збалансованого транспортування: сума пропозиції та попиту однакові.

Σ Пропозиція = Σ Попит

іі) Проблема незбалансованого транспортування: сума пропозиції та сума попиту різні.

Σ Постачання ≠ Σ Попит

У роботі розглянуто два із декількох можливих варіантів розв'язання транспортної задачі: метод "північно-західного кута" та метод "мінімального елемента".

1. Метод "північно-західного кута" полягає в послідовному переборі рядків і стовпців транспортної таблиці, починаючи з лівого верхнього стовпця і верхнього рядка, вписуючи максимальну можливу кількість вантажу у відповідні клітинки так, щоб не було перевищено заявлені в задачі можливості постачальника або споживача. Вартість доставки в даному методі не мають особливого значення, оскільки в подальшому відбувається оптимізація відвантажень. У даному методі розглядається матриця перевезень D, починаючи з верхнього лівого кута. Знаходиться вільна клітинка, в яку записується значення D*ij* = *min*(A*i*, B*j*). Цей результат віднімається від запасів та потреб відповідного складу та магазину. Рядок або стовпчик, який після цієї

операції дорівнюватиме нулю, викреслюється з таблиці. Далі процес повторюється для лівої верхньої клітинки, обираючи із клітинок, що залишилися вільними до того моменту, поки запаси товару не будуть вичерпаними.

Зараз ми подивимося, як застосувати цей дуже простий метод до транспортної проблеми. Ми будемо вивчати етапи цього методу, застосовуючи його в самій задачі.

Компанія з виробництва мобільних телефонів має три філії, розташовані в трьох різних регіонах, скажімо, в Джайпурі, Удайпурі та Мумбаї. Компанія повинна транспортувати мобільні телефони в три місця, скажімо, в Канпур, Пуну і Делі. Доступність у Джайпурі, Удайпурі та Мумбаї становить 40, 60 та 70 одиниць відповідно. Попит у Канпурі, Пуні та Делі становить 70, 40 та 60 відповідно. Вартість транспортування показано в матриці нижче. Використовуйте метод північно-західного кута, щоб знайти базове здійсненне рішення.

Таблиця 1.2 Таблиця розподілу ресурсів

		Пункти призначення			
		Канпур	Пуну	Делі	пропозиція
джерела	Джайпур	4	5	1	40
	Удайпур	3	4	3	60
	Мумбаї	6	2	8	70
		70	40	60	170

Рішення:

Крок 1: Збалансуйте проблему

Збалансувати проблему, тобто нам потрібно перевірити, що якщо;

Σ Пропозиція=Σ Попит

Якщо це так, то ми розглядатимемо дану задачу як збалансовану. А що робити, якщо він не збалансований?

Тобто:

Σ Постачання ≠ Σ Попит

Якщо така умова виникає, ми повинні додати фіктивне джерело або ринок; що робить проблему збалансованою.

Крок 2: Почніть розподіляти з осередку північно-західного кута

Ми почнемо розподіл з лівого верхнього кута (північно-західної) клітинки матриці та зробимо розподіл на основі доступності та попиту.

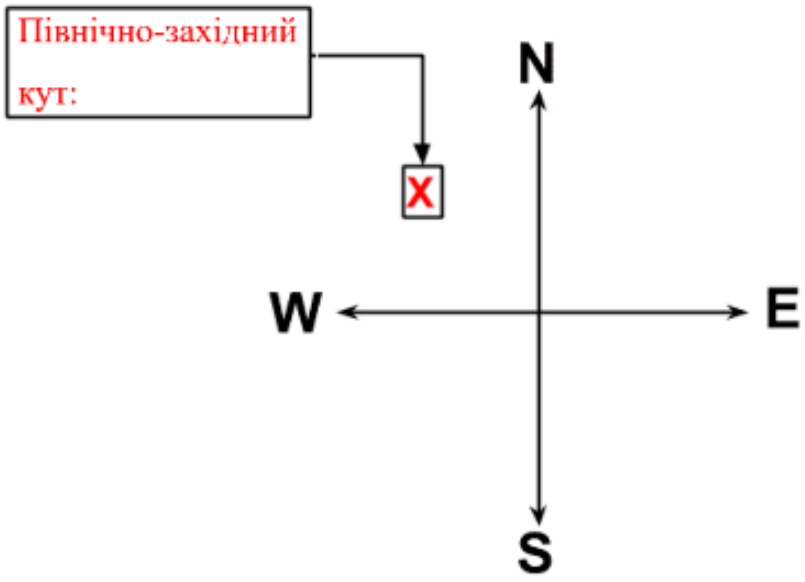


Рисунок 1.2 Розміщення північно-західного кута у системі координат

Тепер перевірте найменшу серед доступності (Supply) і потреби (Demand), що відповідає цій комірці. Найменше значення буде виділено цій комірці і буде перевірено різницю в попиті та пропозиції, що означає, що попит і пропозиція задовольняються, як показано нижче.

Таблиця 1.3 Розподіл ресурсів

		Пункти призначення			
		Каншур	Пуну	Делі	пропозиція
джерела	Джайпур	~~4 (40)~~	~~5~~	~~1~~	~~40~~ ~~0~~
	Удайпур	3	4	3	60
	Мумбаї	6	2	8	70
	попит	~~70~~ 30	40	60	170

Крок 3. Видаліть рядок або стовпець, чия пропозиція чи попит задовольняються, і підготуйте нову матрицю

Оскільки ми виконали доступність або вимогу для цього рядка або стовпця відповідно, видаліть цей рядок або стовпець і підготуйте нову матрицю, як показано нижче.

Таблиця 1.4 Розподіл ресурсів

		Пункти призначення			
		Каншур	Пуну	Делі	пропозиція
джерела	Удайпур	3	4	3	60
	Мумбаї	6	2	8	70
	попит	30	40	60	

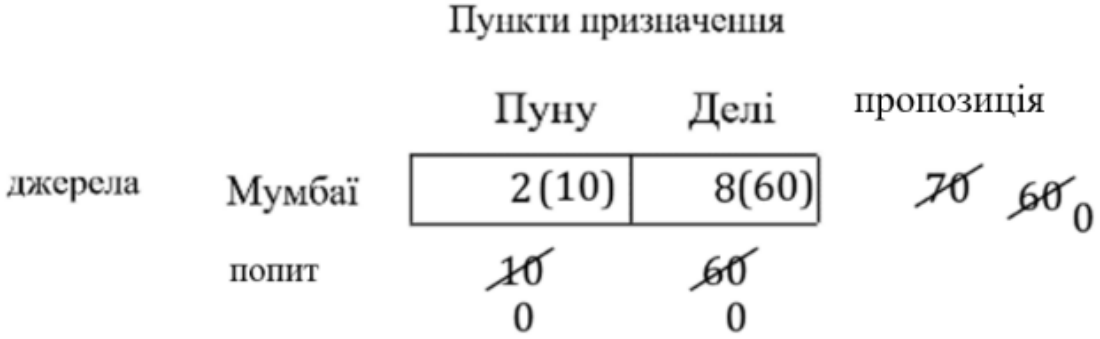
Крок 4: Повторюйте процедуру, поки не закінчаться всі виділення

Повторіть ту саму процедуру виділення нового північно-західного кута, створеного таким чином, і перевірте на основі найменшого значення, як показано нижче, доки всі виділення не закінчаться.

Таблиця 1.5 Розподіл ресурсів

Таблиця 1.6 Розподіл ресурсів

Таблиця 1.7 Розподіл ресурсів



Крок 5: Після того, як усі розподіли закінчилися, запишіть розподіл та розрахуйте вартість транспортування

Після завершення всіх розподілів підготуйте таблицю з позначеними всіма розподілами та розрахуйте вартість транспортування таким чином.

Таблиця 1.8 Розподіл ресурсів

		Пункти призначення			пропозиція
		Каншур	Пуну	Делі	
джерела	Джайпур	4 (40)	5	1	40
	Удайпур	3 (30)	4 (30)	3	60
	Мумбаї	6	2 (10)	8 (60)	70
	попит	70	40	60	

Вартість транспортування:

=(40×40)+(3×30)+(4×30)+(2×10)+(8×60)= 870

2. Метод “мінімального елемента”. Цей метод є більш ефективним, аніж метод “північно-західного куда”. Окрім того, метод “мінімального елемента” зрозумілий та логічний. Його суть полягає в тому, що в транспортній таблиці спочатку заповнюються клітинки з найменшими цінами, а потім — з найбільшими. Тобто кожного разу обираються перевезення із мінімальною можливою вартістю вантажу. У даному методі розглядається вся матриці тарифів, у якій обирається клітинка з найменшим значенням тарифу C*ij*. За наявності декількох клітинок з одинаковими значеннями, обирається будь-яка із них. Далі в клітинку вписується значення D*ij* = *min*(A*i*, B*j*). Отриманий результат віднімається від відповідних запасів та потреб, після чого весь процес виконується повторно, поки увесь запас товарів не буде вичерпано.

Зараз ми спробуємо застосувати цей метод на практиці на тій самій задачі.

Компанія з виробництва мобільних телефонів має три філії, розташовані в трьох різних регіонах, скажімо, в Джайпурі, Удайпурі та Мумбаї. Компанія повинна транспортувати мобільні телефони в три місця, скажімо, в Канпур, Пуну і Делі. Доступність у Джайпурі, Удайпурі та Мумбаї становить 40, 60 та 70 одиниць відповідно. Попит у Канпурі, Пуні та Делі становить 70, 40 та 60 відповідно. Вартість транспортування наведена в матриці нижче.

Використовуйте метод найменших витрат, щоб знайти базове можливе рішення.

Рішення:

Крок 1: Збалансуйте проблему

Збалансувати проблему, тобто нам потрібно перевірити, що якщо;

Σ Пропозиція=Σ Попит

Якщо це так, то ми розглядатимемо дану задачу як збалансовану. А що робити, якщо він не збалансований?

Тобто:

Σ Постачання ≠ Σ Попит

Якщо така умова виникає, ми повинні додати фіктивне джерело або ринок; що робить проблему збалансованою.

Крок 2: Виберіть найнижчу вартість з усієї матриці та розподіліть мінімум попиту чи пропозиції.

Тут ми використовуємо метод найменшої вартості, тому ми будемо визначати найнижче значення комірки у всій цій матриці.

Тут у цій матриці ми маємо 1 (для клітинки: Джайпур-Делі) як найменше значення.

Таблиця 1.9 Розподіл ресурсів

		Пункти призначення			пропозиція
		Каншур	Пуну	Делі	
джерела	Джайпур	~~4~~	~~5~~	~~1~~ (40)	~~40~~ 0
	Удайпур	3	4	3	60
	Мумбаї	6	2	8	70
	попит	70	40	~~60~~ 20	

Отже, рухаючись з цією коміркою, і розподіляючи мінімум попиту чи пропозиції, тобто розподіляючи тут 40 (оскільки значення пропозиції дорівнює 40, тоді як попит дорівнює 60).

Перевіряємо перший рядок, а не останній стовпець, тому що ми виділяємо 40 в осередку для постачання, оскільки це мінімум.

Віднімання виділеної вартості (тобто 40) з відповідного попиту та пропозиції.

Крок 3. Видаліть рядок або стовпець, чия пропозиція чи попит задовольняються, і підготуйте нову матрицю

Оскільки ми виконали попит або пропозицію для цього рядка або стовпця відповідно, видаліть цей рядок або стовпець і підготуйте нову матрицю, як показано нижче:

Таблиця 1.10 Розподіл ресурсів

		Пункти призначення			
		Канпур	Пуну	Делі	пропозиція
джерела	Удайпур	3	4	3	60
	Мумбаї	6	2	8	70
	попит	70	40	20	

Крок 4: Повторіть процедуру

Повторіть ту саму процедуру розподілу найменшого значення в новій згенерованій матриці та перевірте попит або пропозицію на основі найменшого значення (попиту чи пропозиції), як показано нижче, доки не закінчаться всі розподіли.

Таблиця 1.11 Розподіл ресурсів

Таблиця 1.12 Розподіл ресурсів

Таблиця 1.13 Розподіл ресурсів

Крок 5: Після того, як усі розподіли закінчилися, запишіть розподіл та розрахуйте вартість транспортування

Після завершення всіх розподілів підготуйте таблицю з позначеними всіма розподілами та розрахуйте вартість транспортування таким чином:

Таблиця 1.14 Розподіл ресурсів

		Пункти призначення			
		Каннур	Пуну	Делі	пропозиція
джерела	Джайпур	4	5	1 **(40)**	40
	Удайпур	3 **(40)**	4	3 **(20)**	60
	Мумбаї	6 **(30)**	2 **(40)**	8	70
	попит	70	40	60	

Вартість транспортування:

=(1×40)+(3×40)+(3×20)+(6×30)+(2×40)= 480

Варто зазначити, що метод “мінімального елемента” зазвичай дозволяє знайти опорний план за меншу кількість ітерацій, оскільки початковий розподіл ресурсів є більш оптимальним. Тому найбільш доцільно використовувати метод “мінімального елемента” .

2 ПРОГРАМНА РЕАЛІЗАЦІЯ

2.1 Мова програмування C++ та бібліотека Qt 5

Розробником мови C++ є Б’ярн Страуструп. В своїй роботі він спирався на досвід творців мов Симула, Модула 2, абстрактних типів даних. Основні роботи велися в дослідницькому центрі компанії Bell Labs.

Безпосередній попередник C++ — мова Сі з класами - з'явилася в 1979 року, а в 1997 року був прийнятий міжнародний стандарт C++, який фактично підвів підсумки його 20-річного розвитку. Ухвалення стандарту забезпечило одноманітність всіх реалізацій мови C++. Не менш важливим результатом стандартизації став той, що в процесі вироблення і затвердження стандарту мова була уточнена і доповнена рядом існуючих можливостей.

C++ підтримує обєктно-орієнтоване програмування, узагальнене програмування та програмування через шаблони; додаткові типи даних, вставні функції, виключення і т. д. Можливості C++ включають оголошення у вигляді виразів, приведення типів в виді функцій, розширене поняття константності, вставні функції, аргументи за замовчуванням, перевизначення, простори імен (namespace), класи та безліч інших корисних функцій.

Для проекту було обрано бібліотеку Qt — кросплатформленний фреймворк, який значно розширює не тільки стандартну бібліотеку C++, а й додає нові можливості, як-от створення графічних застосунків.

Основною особливістю Qt від інших бібліотек є е Meta Object Compiler (MOC) — попередньої системи обробки вихідного коду. MOC дозволяє набагато збільшити можливості звичайних бібліотек, вводячи такі поняття як слоти та сигнали, які використовуються для комунікації між об’єктами. Сигнал виробляється, коли трапляється певна подія, а слот — це функція, яка викликається після спрацьовування сигналу. Окрім того, це дозволяє зробити код більш лаконічним. Утиліта MOC шукає у файлах вихідного коду макрос

Q_OBJECT та створює додатковий файл (.moc), який вміщує мета-об’єктний код.

Qt надає програмісту не тільки зручний набір бібліотек класів, а й певну модель розробки програм, певний каркас їх структури. Дотримання принципів і правил “хорошого стилю програмування на C++/Qt” значно зменшує таких частих помилок як витік пам’яті (memory leaks), необроблені винятки, незвільнені дескріптори ресурсних об’єктів, чим нерідко страждають програми, написані за допомогою стандартної бібліотеки C++.

Важливою перевагою Qt є добре продуманий, логічний набір класів із дуже високим рівнем абстакції. Завдяки цьому програмістам, які використовують Qt, потрібно писати значно менше коду, аніж під час використання, наприклад, класів MFC. Сам код виглядає простіше, логічніше та зрозуміліше.

Qt комплектується візуальним середовищем розробки графічного інтерфейсу “Qt Designer” для створення графічного інтерфейсу з максимальною ефективністю та простотою.

В складі Qt також є “Qt Linguist” — графічна утиліта, яка дозволяє спростити локалізацію програм на різні мови. В самому вихідному коді не потрібно вказувати переклад кожного рядка, що б сильно збільшувало об’єм коду, достатньо всього лиш вказати ключове слово “tr”, яке означатиме, что наступний текст потрібно буде перекласти на іншу мову.

Також завдяки добре стуктурованій документації в утиліті “Qt Assistant” з’являється можливість переглянути інформацію для потрібного класу, ознайомитися з прикладами.

В основному Qt використовується для створення додатків з графічним інтерфейсом, перевагою яких є кросплатформленість.

Навіть якщо програмісту в даний конкретний момент кросплатформленість не потрібна (наприклад, планується версія тільки для Windows), то у випадку, якщо необхідність підтримки інших операційних

систем з’явиться завтра, то для цього необхідно просто скомпілювати прогрму для потрібної системи.

У випадку використання, наприклад, MFC, знадобиться багато часу, щоб перенести вихідний код.

Qt є повністю об’єктно-орієнтованим, легко розширюваним та з підтримкою техніки компонентного програмування. Основою побудови графічних інтерфейсів є віджети. Кожен елемент у вікні програми, як-от кнопка або перемикач, є окремим незалежним віджетом. Віджети генерують різні сигнали, які, в свою чергу, можуть викликати слоти — спеціальні функції, які реагують на різні події, які відбуваються на вікні.

Багато компаній-розробники додатків Windows використовують Qt ще з однієї причини: навіть якщо код пишеться і в доступному для огляду майбутньому буде писатися тільки для платформи Windows і тестується тільки на ній, можливість скомпілювати один і той же вихідний код на одній і тій же платформі Windows двома різними компіляторами (Microsoft Visual C ++ I GCC / Win32) гарантує кращу якість вихідного коду і кращу його сумісність зі стандартом C ++, що важливо для коду, який планується довго підтримувати і розвивати. У бібліотеки Qt існують “прив'язки” до багатьох популярних мов програмування: Python, Java і іншим. Ці «прив'язки» дозволяють багатьом програмістам писати програми з графічним інтерфейсом, заснованим на Qt на улюбленому мовою, яка не вдаючись в усі подробиці C++.

2.2 Середовище розробки Qt Creator

Qt Creator — кросплатформленна вільна IDE для розробки на C, C++ та QML з можливістю створення віджетів та їх редагування безпосередньо в редакторі. Основним завданням середовища розробки є спрощення розробки програм на різних платформах

Починаючи з версії 4.5.0 в комплект Qt добавлено візуальне середовище розробки Qt Creator, яка включає в себе редактор коду, справку, компілятор,

можливість відлагоджувати програму та інше. Автодоповнення тексту дозволяє значно пришвидшити розробку програми, а виділення (підсвітка) коду дозволяє покращити читання коду. Також можна вказати власний стиль вирівнювання, відступів, способу форматування дужок, а також підтримка сигнатур методів. Крім того, інтерфейс середовища передбачає вбудовану систему допомоги розробникам QtAssistant і спеціальний дизайнер QtDesigner, які допоможуть швидко і без помилок скласти текст програми і створити її інтерфейс засобами самого Qt, тим самим виключаючи можливі помилки при додаванні елементів на форму. Ще одна важлива перевага Qt Creator — це його поширення під ліцензією LGPL, що дозволяє створювати власні комерційні проекти без будь-якого винагороди виробникам середовища.

Отже, Qt Creator IDE — це повне кросплатформленне рішення для розробників, що дозволяє комфортно створювати додаток для декількох операційних систем, мобільних пристроїв і вбудованих платформ. До того ж ПО пропонує інструменти та функції, які допомагають розробників писати код, знижувати ризики проектів і підвищувати продуктивність праці.

2.3 Технічне завдання

Створення програмного продукту для вивчення дисципліни “Дослідження операцій” і таких її розділів, як табличний симплекский метод та транспортна задача.

Користувачам будем запропоновано як автоматичний режим розв’язування поставлених задач, так і ручний, де кожен крок перевірятиметься комп’ютером і на основі дій користувача надаватимуться певні інструкції та поради на кожному етапі розв’язання задачі. Таким чином вдасться донести потрібні та доречні поради саме в ті моменти, коли допомога справді потрібна. Завдяки такому підходу в учня не виникне жодних проблем з розв’язанням будь-якої задачі — це саме та річ, яка закладена у всю

програму: достатньо усього лиш ознайомитися з порадами та сумлінно їх дотримуватися.

2.3.1 Модель задачі

У симплексному методі необхідно розробити методи для:

— зведення початкової задачі до канонічного вигляду, де за необхідності додати додаткові або штучні змінні;
— побудови початкової матриці коефіцієнтів;
— побудови початкової симплексної таблиці;
— пошук опорного рядка та стовпчика, враховуючи певні нюанси задачі, як-от її виродженість;
— розрахунок нової таблиці;
— побудова детальних коментарів для користувача щодо ходу розв’язування задачі;
— реалізація графічного інтерфейсу з візуальними підказками та короткими gif-інструкціями;
— перевірка дій користувача та їх візуальне виділення під час неправильних кроків для подального виправлення та побудови підказок.

У транспортній задачі необхідно розробити наступні методи:

— розробка алгоритму для розв’язання задачі методом мінімального елемента та методом північно-західного кута;
— зведення початкової задачі до закритого типу;
— пошук потенціалів для покращення поточного результату;
— пошук циклу транспортної задачі;
— пошук всіх можливих розв’язків конкретної задачі для відображення найбільш ефективного розв’язку;
— розробка алгоритму для перевірки дій користувача під час заповнення таблиці ресурсів: цей алгоритм відштовхується від дій користвуча, що дає змогу максимально правильно перевірити кожен крок;

— розробка алгоритму для перевірки побудованих користувачем потенціалів та циклу;

— розробка графічного інтерфейсу для зручного введення даних задачі та подальшого розв’язання із зображенням інстукцій та порад на кожному етапі.

2.3.2 Організація вхідних і вихідних даних

Вхідні дані для розв’язування симплекс-методу подаються з графічної частини інтерфейсу, які зображені у вигляді таблиць, де кожен рядок є окремим рівнянням. Далі надана інформація конвертується в структуру, зрозумілу для комп’ютера з наступними полями: числові аргументи, значення додаткової змінної (1, -1 або 0, якщо немає), логічний знак та значення вільного члена. Така структура називається Equation.

Цільова ж функція зберігається окремо у структурі ObjectiveFunction та відрізняється від рівняння тим, що зберігає в собі тип задачі: максимізація або мінімізація.

Вихідні дані зберігаються у структурі Simplex::SimplexTable, у якій окремо зберігаються базисні та небазисні змінні, тип задачі, та поточний опорний рядок та стовпчик, а також поточну симплексну таблицю. Ці дані передаються у графічний інтерфейс для представлення їх в зручному для користувача виді.

Вхідні дані для розв’язування симплекс-методу подаються з графічної частини інтерфейсу, які зображені у вигляді таблиць розмірності N + 1 x M + 1 для того, щоб останній рядок і стовпчик використовувався для введення кількості потреб та запасів відповідно.

Для автоматичного розв’язання спочатку необхідно отримати початковий опорний план розподілу ресурсів. Для цього достатньо використати статичну функцію створеної бібліотеки AllMinimumElementPlans::build та передати наступні параметри: таблицю тарифів, вектор потреб, вектор запасів, а також алгоритм розподілу ресурсів: методом мінімального елемента або північно-західного кута. Щоб отримати

розв’язок задачі необхідно використати статичний метод TransportationProblem::solve(), передавши як аргумент функції щойно побудовану таблицю та матрицю тарифів.

Вихідні дані, тобто результат роботи програми, зберігаються у структурі TransportationProblem::Table, поля якої вміщують значення потенціалів до поточної таблиці, цикл, а також саму таблицю.

2.4 Алгоритм розв’язання поставлених задач

2.4.1 Симплекс-метод

Початок розв’язування задачі полягає в передачі вхідних даних обмежень у вигляді вектора, а також цільової функції у клас Simplex та подальшому використанні метода solve() для отримання розв’язку.

Розв’язок починається з виклику метода для зведення обмежень до канонічного вигляду. У циклі спочатку перевіряється знак нерівності та вільний член. Якщо знак рівності дорівнює логічному знаку “=” та вільний член додатний, то така рівність вважається канонічною. Інакше вважається, що знак нерівності не дорівнює логічному знаку “=” і таку нерівність потрібно звести до канонічного вигляду. Перед цим необхідно перевірити, чи вільний член від’ємний та за потреби домножити кожен аргумент на мінус один (-1), тобто змінити знак на протилежний. Після зведення всіх обмежень до канонічного вигляду необхідно перевірити, чи цільова функція максимізована та, у випадку потреби, максимізувати її, змінивши знак біля кожного аргумента на протилежний (домножити на мінус один (-1)).

Наступним кроком є побудова матриці коефіцієнтів функцією createMatrixCoefficients().

Таблиця коефіцієнтів – це значення при всіх змінних, які входять до кожного рівняння, враховуючи додаткові змінні, а також штучні змінні для створення одиничної матриці.

Побудова матриці коефіцієнтів починається із попереднього обчислення її початкової розмірності (кількості стовпчиків) за наступною формулою: кількість аргументів при обмеженні або цільовій функції додати кількість ненульових додаткових змінних — вони використовуватимуться для одиничної матриці.

Далі циклічно необхідно перевірити, чи в матриці знаходиться повний набір одиничних стовпчиків та, за необхідності, додати в рядки, де їх не вистачає. Така задача називатиметься задачею зі штучним базисом: змінна типу bool із назвою artificialBasic отримає логічне значення true.

На наступному етапі потрібно побудувати першу симплексну таблицю та обчислити рядок функціоналу (значення рядка оцінок) за допомогою функції calculateFreeMemberAndRatingValues().

Після використання цієї функції буде отримано таблицю N x M, де останній стовпчик є вільними членами, а останній рядок — рядком оцінок.

Варто зазначити, що під час розв’язування задач зі штучним базисом, усі значення аргументів необхідно сприймати за нуль. В момент, коли в симплексній таблиці рядок оцінок буде із додатніми значеннями, необхідно вивести штучні змінні із таблиці, видаливши стовпці, в рядках оцінок яких знаходиться значення 1 (один). Таким чином задача із штучного базису перетвориться у звичайну.

Після побудови початкової симплексної таблиці починається безпосередній ітераційний розв’язок задачі. Алгоритм зображено на блок-схемі.

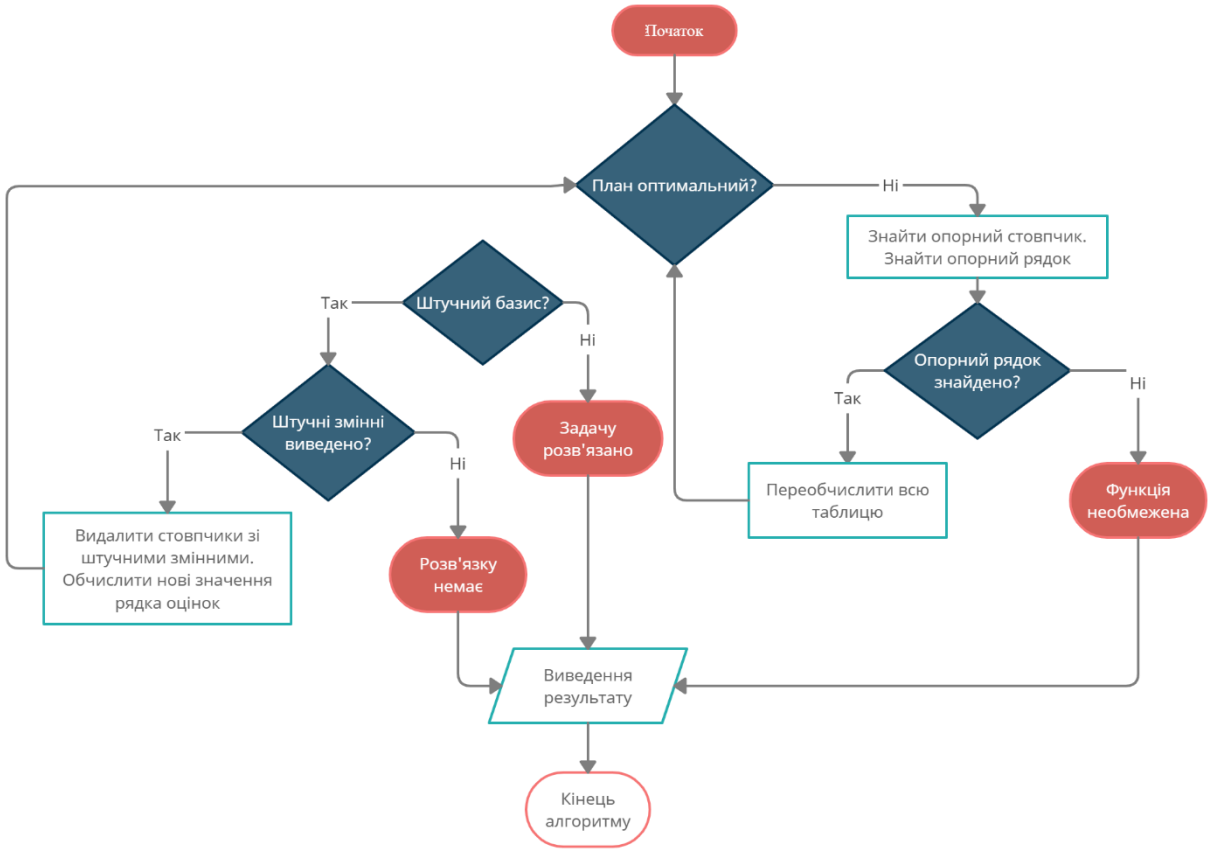


Рисунок 2.1 - Алгоритм ітераційного розв’язку задачі

Після побудови усіх табиць їх можна отримати за допомогою методу Simplex::getSimplexTables(), яка повертає векторс туктур Simplex::SimplexTable.

Далі отриманий результат необхідно зобразити у графічному інтерфейсі у відповідних таблицях.

Для відображення графічного інтерфейсу використовуються об’єкти, успадковані від класу QWidget. Віджетами називаються графічні об’єкти-контейнери, які використовуються для зберігання та відображення візуальних об’єктів. Кожен візуальний об’єкт обов’язково наслідується від батьківського класу QWidget. Разом із симплексними таблицями користувач отримує візуальне представлення обмежень у канонічній формі, а також матрицю коефіцієнтів.

Під час ручного розв’язання додається тільки алгоритм перевірки дій користувача, тобто методи для порівняння очікуваного результату із поточним. Для цього було реалізовано наступні методи:

— перевірка правильності побудови канонічної форми кожного із обмежень. На цьому етапі перевіряються усі аргументи, логічний знак та вільний член на їх відповідність очікуваному правильному результату і, у випадку, невідповідності, створюється коментар із відповідним текстом, який вказує на тип помилки та її подальше графічне виділення кольором. Таким чином користувчеві буде більш зрозуміло, де потрібно виправити допущену помилку.

— перевірка правильності побудови матриці коефіцієнтів. Користувачеві запропоновано додати необхідні штучні змінні, якщо це необхідно для побудови одиничної матриці.

— перевірка правильності побудови кожної із симплексних таблиць.

— перехід до заповнення наступної таблиці, якщо задача не розв’язана.

2.4.2 Транспортна задача

Автоматичний режим розв’язування транспортної задачі починається з розподілення ресурсів, використовуючи певний алгоритм: метод мінімального елемента або північно-західного кута.

Далі використовується функція для зведення задачі до закритого типу: обчислюється сума запасів та потреб, та за необхідності додається фіктивний рядок або стовпчик.

Оскільки розподілити ресурси можна різними способами (наприклад, викреслювати не рядок, а стовпчик і навпаки), то таких варіантів можливого розподілу існує декілька. У таких випадках необіхно побудувати усі можливі варіанти такого розподілу, розв’язати задачі з кожним із варіантів та відсортувати їх від найефективнішного до менш ефективних методів.

Наступним кроком є ітераційне покращення початкового опорного плану методом потенціалів. Алгоритм методу потенціалів полягає в таких етапах:

— визначення типу задачі: відкрита або закрита. На етапі побудови потенціалів програмою гарантується, що задача закритого типу;

— перевірка задачі на виродженість, тобто сума базисних клітинок обов'язкова повинна дорівнювати N + M – 1. Програмою гарантується, що всі можливі побудовані плани відповідають цій умові.

У методі потенціалів кожному рядку i і кожному стовпцю j транспортної таблиці ставляться у відповідність числа (потенціали) u_i і v_j. Для кожної базисної змінної x_{ij}, потенціали u_i і v_j задовольняють рівнянню: $u_i + v_i = c_{ij}$.

Щоб знайти значення потенціалів з цієї системи рівнянь, потрібно присвоїти одному з них довільне значення (зазвичай вважають $u_1 = 0$) і потім послідовно обчислювати значення інших потенціалів.

Далі, використовуючи знайдені значення потенціалів, для кожної небазисной змінної обчислюються величини $u_i + v_i = c_{ij}$.

Після побудови потенціалів необхідно перевірити план на оптимальність. План вважається оптимальним, якщо сума $u_i + v_i$ по всіх небазисних клітинках є більшим або рівним значенню відповідного тарифу. Таких клітинок може бути декілька. Для кращого результату обирається та, у якої значення $u_i + v_i - \text{tariff}_{ij}$ є максимальним.

Наступний етап полягає в покращенні опорного плану. Для цього, відштовхуючись від щойно знайденого потрібного елемента, будується цикл транспортної задачі. Циклом називається послідовне з'єднання базисних клітинок не більше двох в рядку або стовпчику, причому тільки перший елемент циклу є небазисним. Оскільки цикл в пам'яті ком'ютера зберігається в виді вектора, то усі елементи з парними індексами потрібно вважати зі знаком "плюс" в циклі, а непарні — зі знаком "мінус".

Побудувавши цикл, необхідно покращити поточний план. Для цього виконується пошук мінімального значення тарифа серед клітинок зі знаком

“мінус”. Далі по елементах циклу в таблиці потрібно додати або відняти відповідне знайдене мінімальне значення тарифу.

Ці дії є ітераційними та виконуються циклічно, поки план є неоптимальним. Цикл розв’язання можна зобразити наступною блок-схемою:

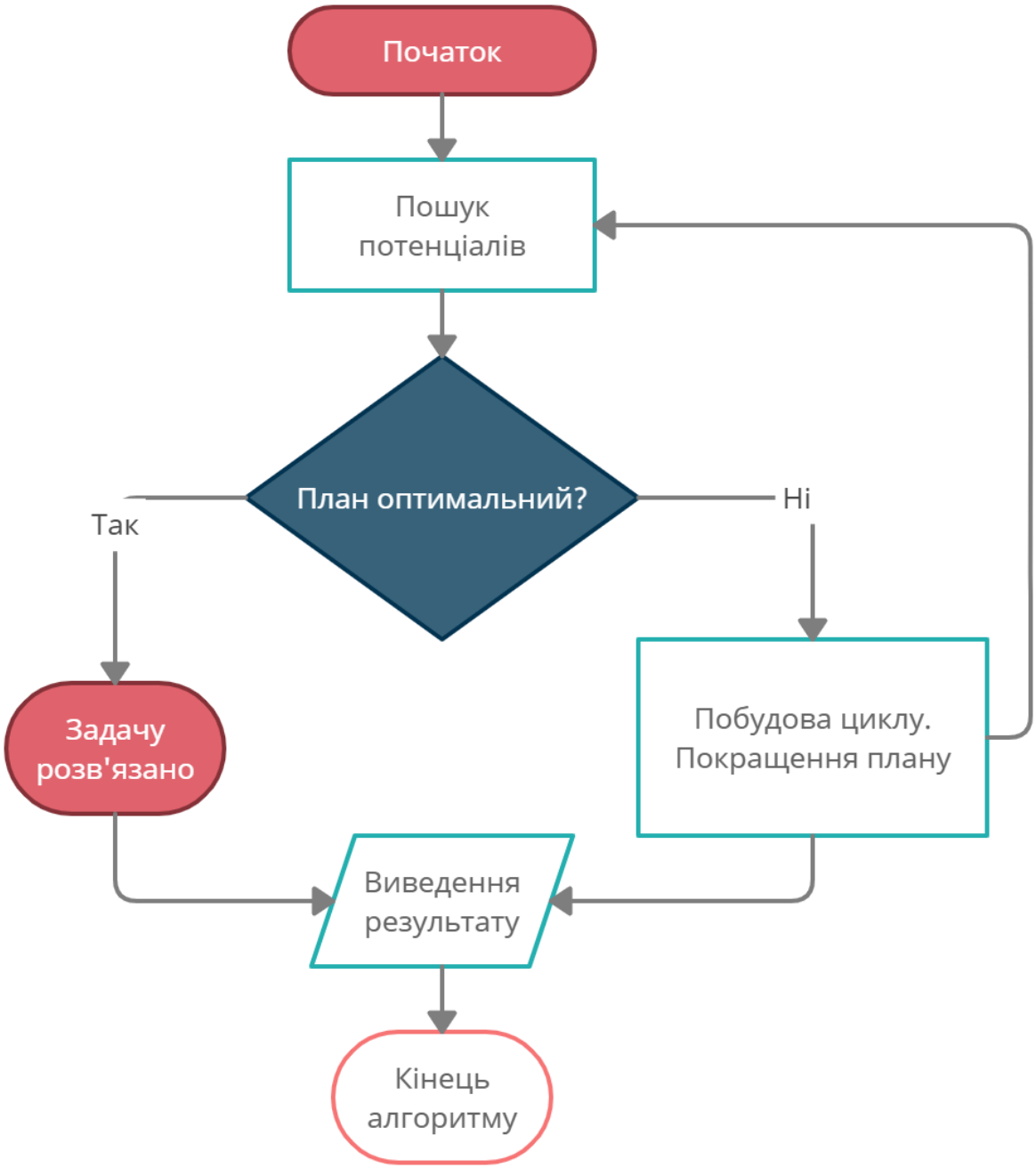


Рисунок 2.2 - Цикл розв’язання транспортної задачі

Ручний режим розв’язання полягає в побудові алгоритмів, які на основі кроків (вибору клітинок) користувачем будують власний розв’язок, який далі порівнюється із варіантом користувача. На основі усіх можливих побудованих планів обирається той, який є найбільших схожим до порівнювального варіанту та у випадку невідповідності користувачеві буде вказано на помилку, якщо, наприклад, замість рядка необхідно викреслити стовпчик.

2.5 Алгоритм шифрування AES

2.5.1 Передумови та історія створення Advanced Encryption Standard

У 80-х роках основним симетричним криптоалгоритмом для внутрішнього застосування США був DES (Data Encryption Standard). Проте вже у 90-х роках стали виявлятися його основні недоліки. Головним у тому числі була довжина ключа, що становить 56 біт. Такий розмір ставав недостатньо великим через постійне зростання продуктивності ЕОМ, оскільки ключ міг бути розкритий методом нормального перебору всіх можливих варіантів шифрування.

З цих причин у 1997 році NIST (National Institute of Standards and Technology) оголосив конкурс на новий стандарт симетричного криптоалгоритму. Було встановлено такі обов'язкові вимоги до кандидатів:

- 128-бітний розмір блоку даних, що шифруються;
- Не менше 3 розмірів ключів (128, 192, 256 біт), які мають підтримуватись алгоритмом;
- Використання операцій, що легко реалізуються і апаратно, і програмно;
- Простота структури шифру, щоб заінтересовані сторони могли самостійно проводити криптоаналіз;

Переможцем цього конкурсу, названого AES – Advanced Encryption Standard, став алгоритм Rijndael (далі алгоритм AES). Він продемонстрував стійкість до атак, відносно низький рівень енергоспоживання та відносно

невеликий час роботи. З іншого боку, алгоритму властивий внутрішній паралелізм, з цього ефективно використовуються процесорні ресурси, і додатково зменшується час роботи алгоритму.

Таблиця 2.1 - Порівняння AES та його попередника – DES

	DES	AES
Розроблений	1997	200
Тип шифру	Симетричний блоковий шифр	Симетричний блоковий шифр
Розмір блоку	64 біта	128 біт
Довжина ключа	56 біт	128/192/256 біт
Безпека	Став небезпечним	Вважається безпечним

AES — це шифр, метод шифрування та дешифрування інформації. Щоразу, коли ви передаєте файли через безпечні протоколи передачі файлів, такі як HTTPS, FTPS, SFTP, WebDAVS, OFTP або AS2, є велика ймовірність, що ваші дані будуть зашифровані за допомогою шифрів AES — AES 256, 192 або 128. Незабаром ми обговоримо докладніше про ці шифрування AES.

Різне програмне забезпечення для безпечної керованої передачі файлів може бути оснащено різними алгоритмами шифрування. Деякі шифри можуть бути включені в певні вибірки, але відсутні в інших. Не AES. AES майже завжди буде присутній у всіх, крім кількох. Чому це так? Все почалося, коли уряд США почав шукати новий алгоритм шифрування, який можна було б використовувати для захисту конфіденційних даних.

Приблизно два десятиліття з 1977 року уряд США використовував шифр під назвою DES (Стандарт шифрування даних) для захисту конфіденційної, несекретної інформації. На жаль, пізніше було доведено, що цей шифр не є надійним, що змусило уряд шукати заміну.

Це призвело до процесу стандартизації, який залучив 15 конкуруючих проектів шифрування, які включали, серед іншого, MARS від IBM, RC6 від RSA Security, Serpent, Twofish і Rijndeal. Саме Rijndael, розроблений двома бельгійськими криптографами (Джоан Демен і Вінсентом Рейменом), з часом став стандартом і відомий як Advanced Encryption Standard або AES.

Процес відбору був дуже суворим, тривав п’ять років. Протягом цього періоду багато експертів із криптографічного співтовариства провели детальні тести та копіткі обговорення, щоб знайти вразливі та слабкі місця. Участь різних секторів, яка продемонструвала відкритість процесу відбору, говорить про те, наскільки цей процес був довірливим.

Хоча міцність шифру проти різних атак була головним фактором при виборі стандарту, він включав інші фактори, такі як швидкість, універсальність і вимоги до обчислень. Уряд хотів, щоб стандарт шифрування був не тільки міцним, але й швидким, надійним і легко реалізованим як у програмному, так і в апаратному забезпеченні — навіть у тих, у яких обмежений процесор та пам’ять.

Незважаючи на те, що інші алгоритми шифрування також були дуже хорошими, шифр Рейндаля в кінцевому підсумку був обраний і оголошений NIST (Національним інститутом стандартів і технологій) у 2001 році як федеральний стандарт обробки інформації або стандарт FIPS. Його схвалив міністр торгівлі, а потім визнано стандартом федерального уряду наступного року.

На цьому розвиток AES не закінчився. У 2003 році уряд визнав його придатним для захисту секретної інформації. АНБ (Агентство національної

безпеки) все ще використовує AES для шифрування абсолютно секретної інформації.

Саме тому AES завоювала довіру в різних галузях. Якщо це досить добре для АНБ, то воно має бути достатньо добре для бізнесу.

2.5.2 Як працює AES

AES належить до сімейства шифрів, відомих як блокові шифри. Блоковий шифр - це алгоритм, який шифрує дані на основі кожного блоку. Розмір кожного блоку зазвичай вимірюється в бітах. AES, наприклад, має довжину 128 біт. Це означає, що AES працюватиме з 128 бітами відкритого тексту для створення 128 біт шифрованого тексту.

Як і майже всі сучасні алгоритми шифрування, AES вимагає використання секретних ключів під час процесів шифрування та дешифрування. AES підтримує три ключі з різною довжиною: 128-розрядні ключі, 192-розрядні ключі та 256-бітні ключі. Розмір ключа також важливий. Чим довший ключ, тим міцніше шифрування. Отже, шифрування AES 128 найслабше, а шифрування AES 256 найсильніше.

Що стосується продуктивності, то коротші ключі призводять до швидшого шифрування в порівнянні з довшими ключами. Отже, 128-розрядне шифрування AES швидше, ніж 256-розрядне шифрування AES.

Ключі, що використовуються в шифруванні AES, є тими самими ключами, що використовуються при розшифруванні AES. Коли під час шифрування та дешифрування використовуються одні й ті ж ключі, алгоритм називається симетричним.

Алгоритм AES використовує підстановку-перестановку, або мережу SP, з кількома раундами для створення зашифрованого тексту. Кількість раундів залежить від використовуваного розміру ключа. Розмір ключа 128 біт диктує десять раундів, 192-бітовий розмір ключа диктує 12 раундів, а розмір ключа 256 біт має 14 раундів. Для кожного з цих раундів потрібен ключ раунду, але

оскільки в алгоритм вводиться лише один ключ, цей ключ потрібно розширити, щоб отримати ключі для кожного раунду, включаючи раунд 0.

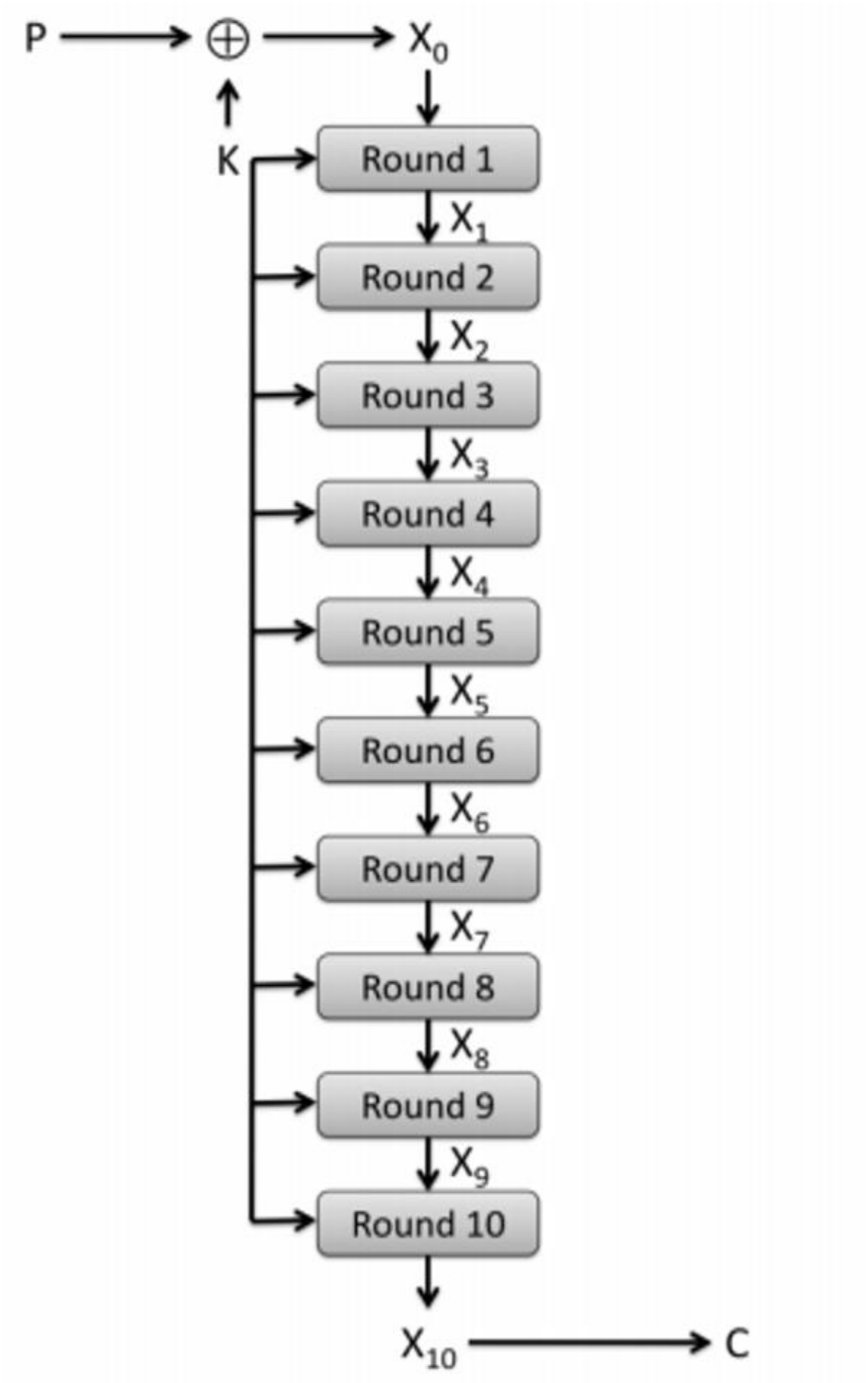


Рисунок 2.3 – Смехатичне зображення алгоритму AES

Кожен раунд в алгоритмі складається з чотирьох кроків.

1. Підміна байтів

На першому кроці байти блокового тексту замінюються на основі правил, продиктованих заздалегідь визначеними S-боксами (скорочено від поля підстановки).

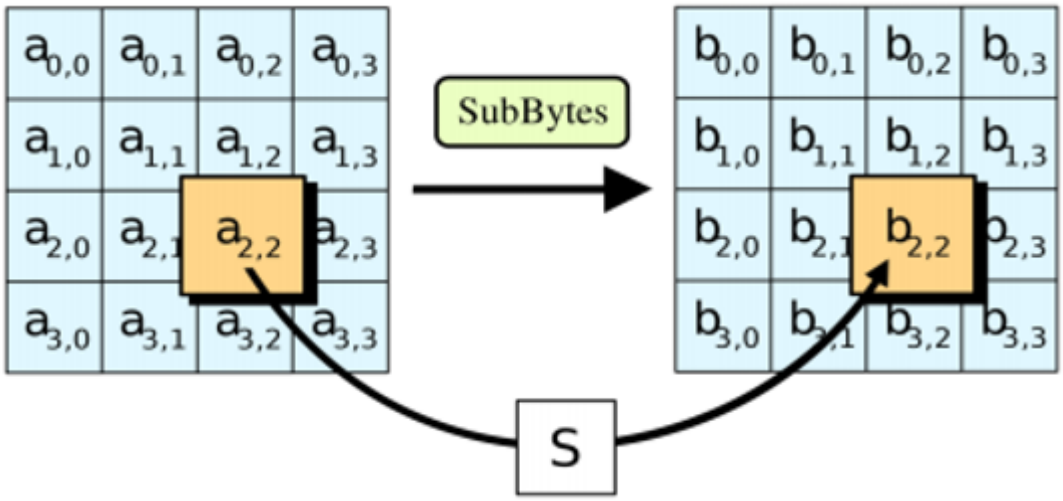


Рисунок 2.4 – Перестановка байтів

2. Зміщення рядків

Далі йде крок перестановки. На цьому кроці всі рядки, крім першого, зсуваються на один, як показано нижче.

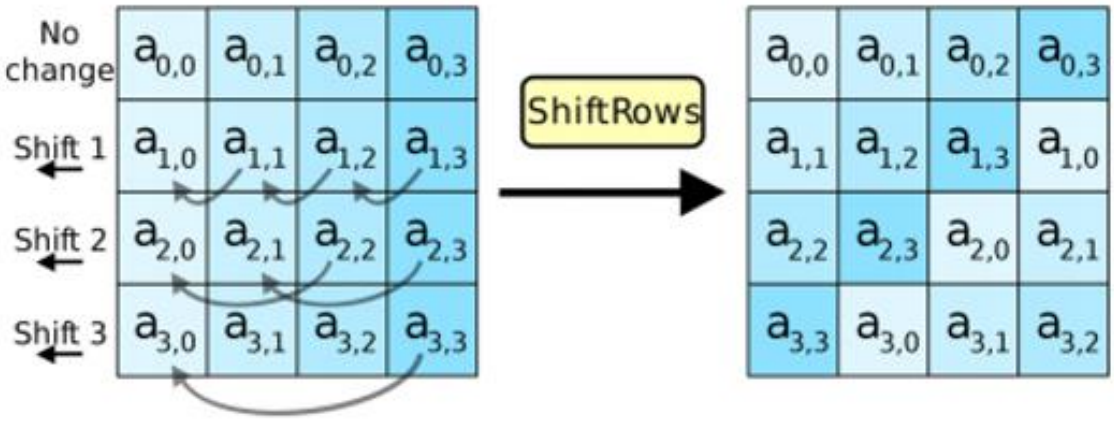


Рисунок 2.5 – Перестановка рядків

3. Змішування стовпчиків

На третьому кроці використовується шифр Хілла, щоб більше змішувати повідомлення шляхом змішування стовпців блоку.

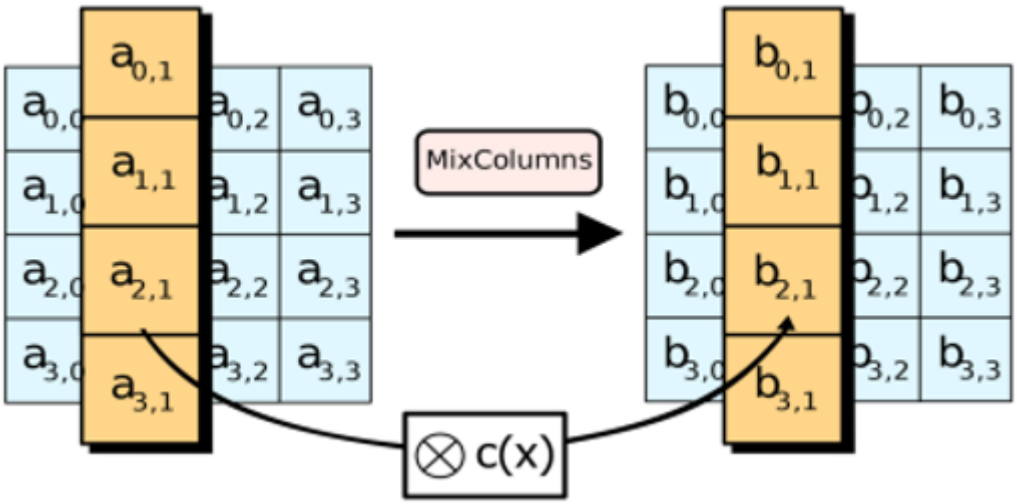


Рисунок 2.6 – Змішування стовпчиків

4. Додавання круглого ключа

На останньому кроці повідомлення перетворюється на XOR за допомогою відповідного круглого ключа.

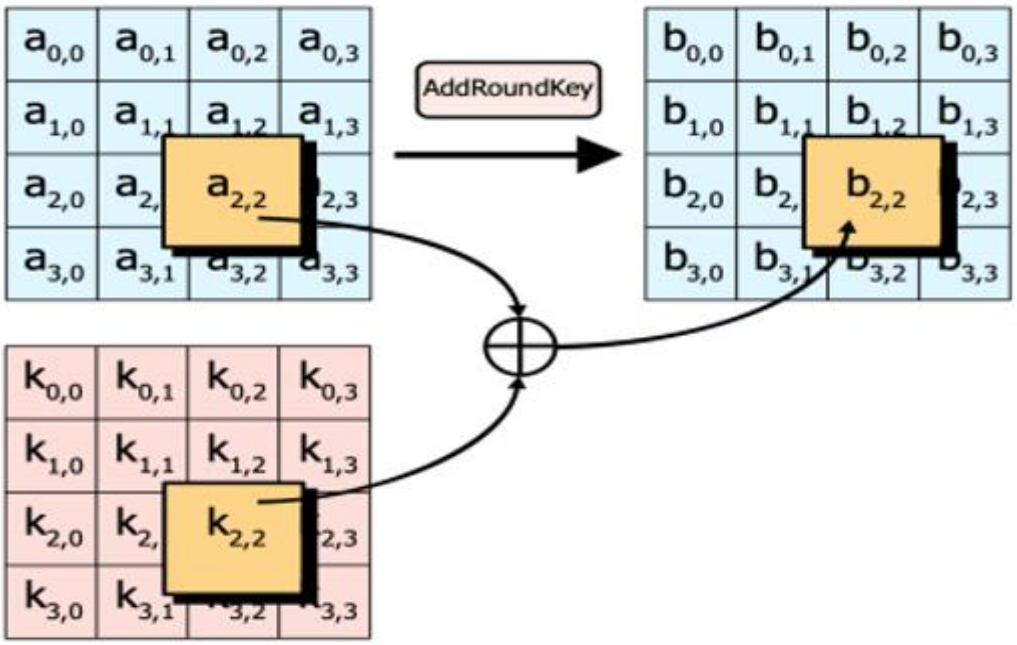


Рисунок 2.7 – Додавання ключа

При повторному виконанні ці дії забезпечують безпеку остаточного зашифрованого тексту.

2.5.3 Атаки на AES

Дослідники постійно намагаються зламати AES за допомогою життєздатних методів. Причина, чому дослідники намагаються зламати AES, полягає в тому, щоб бути на крок попереду зловмисників. Якби зловмисник зламав AES і тримав це в секреті, світ продовжував би використовувати AES, вважаючи, що він повністю безпечний. Наразі було запропоновано кілька різних теоретичних атак, зокрема:

Атака за пов'язаним ключем: Атака за пов'язаним ключем передбачає визначення того, як шифр працює під різними ключами. Ця техніка криптоаналізу передбачає подачу шифру, який використовується для шифрування даних, кількох різних ключів з одним відкритим текстом. Процес, який відбувається між ключем і шифром, може допомогти визначити математичний зв’язок між шифром і ключем, таким чином допомагаючи визначити фактичне значення ключа. Однак цей метод атаки не вважається великою загрозою для AES, оскільки він марний, якщо протоколи реалізовані правильно.

Атака «Відмінний ключ»: успішно використана атака, яка використовувала відомий ключ для з’ясування внутрішньої роботи алгоритму AES-128 із 8 раундів. Оскільки це було зроблено за алгоритмом 8 раундів, на відміну від офіційного алгоритму з 10 раундами, це атака, яка не повинна викликати проблем з будь-якими офіційними алгоритмами AES.

Атака по бічним каналам: Атака з бічного каналу передбачає витік інформації з інфраструктури організації. Дані просочуються через місця, і зловмисник прослуховує звук, інформацію про час, електромагнітну інформацію або споживану потужність, щоб отримати висновки від

алгоритму, які потім можуть бути використані для його зламу. Однак це можна зупинити, усунувши джерело витоку або переконавшись, що в інформації про витік немає шаблону.

Компроміс ключа: хоча це не є прямою атакою на алгоритм AES, компроміс ключа, який використовується для шифрування, ламає весь алгоритм AES. Ось чому належне керування ключами та безпека є суттєво важливими для IT-інфраструктури будь-якої організації.

Квантові обчислення: Квантові обчислення є наступниками класичних обчислень, якими ми зараз займаємося, які все ще знаходяться в процесі створення та розуміння. Хоча це ще не повністю реалізовано, створення квантових комп'ютерів зробить всю класичну обчислювальну криптографію неактуальною, оскільки квантові обчислення можуть зламати будь-який класичний криптографічний алгоритм за потенційно лічені секунди.

2.5.4 Режими AES

Алгоритм AES має три режими роботи, а саме:

1) Режим електронної кодової книги (ECB).
2) Режим ланцюжка блоків шифрування (CBC).
3) Режим лічильника (CTR).

ECB (скорочення від електронної кодової книги) є найпростішим режимом блочного шифрування AES. У режимі ECB блок B_i шифрується за такою формулою:

$$C_i = E_k(B_i) \tag{2.1}$$

де E_K позначає алгоритм блочного шифрування з використанням ключа K, а C_i — шифр, що відповідає B_i.

Розшифровка за допомогою режиму ECB також проста за такою формулою:

$$B_i = D_K(C_i) \tag{2.2}$$

де D_K позначає алгоритм дешифрування блоку за допомогою ключа K.

Найбільш очевидна перевага використання режиму ECB полягає в тому, наскільки він спрощений. Інша головна перевага полягає в тому, що ЄЦБ може терпіти втрату блоків, не впливаючи на інші доступні блоки. Ця перевага актуальна у випадку, коли блоки надсилаються по мережі у вигляді пакетів. Ця стійкість стає можливою завдяки тому, що будь-який блок Bi не залежить від жодного із сусідніх блоків.

Незважаючи на свої переваги, на ECB дивляться зневажливо через те, що алгоритм шифрування є повністю детермінованим. Простіше кажучи, ідентичні блоки матимуть однакові шифри в режимі ECB, що може виявити шаблони, які мають блоки; тож ЄЦБ повністю не приховує своїх деталей. Це загроза безпеці його користувачів.

CBC (скорочення від ланцюжка шифр-блок) — це режим блочного шифрування AES, який перевершує режим ECB у приховуванні шаблонів у відкритому тексті. У режимі CBC це досягається шляхом вимкнення першого блоку відкритого тексту (B1) з вектором ініціалізації перед його шифруванням. CBC також включає ланцюжок блоків, оскільки кожен наступний блок відкритого тексту виконується XOR із зашифрованим текстом попереднього блоку.

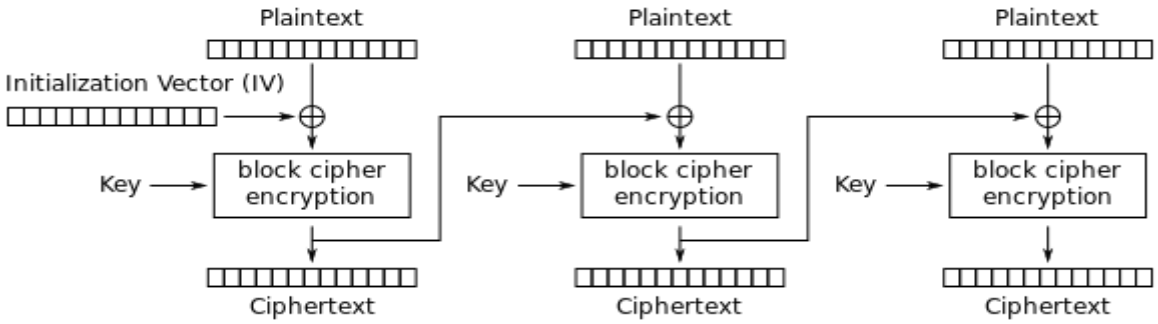


Рисунок 2.8 – Схема шифрування в режимі CBC

Якщо узагальнити цей процес у формулу, він буде виглядати так:

$$C_i = E_K(B_i \oplus C_{i-1}) \quad (2.3)$$

де E_K позначає алгоритм блочного шифрування з використанням ключа *K*, а C_{i-1} є шифром, що відповідає $B_{i-1.}$

Аналогічно, розшифрування за допомогою CBC можна виконати за допомогою:

$$B_i = D_K(C_i) \oplus (C_{i-1}) \quad (2.4)$$

де D_K позначає алгоритм дешифрування блоку за допомогою ключа *K*.

Найбільша перевага CBC над ECB полягає в тому, що в режимі CBC однакові блоки не мають однакового шифру. Це тому, що вектор ініціалізації додає випадковий коефіцієнт до кожного блоку; отже, чому ті самі блоки в різних позиціях будуть мати різні шифри.

Хоча режим CBC є більш безпечним, його шифрування не толерантно до втрат блоку. Це тому, що блоки залежать від своїх попередніх блоків для шифрування. Отже, якщо блок Bi буде втрачено, шифрування всіх наступних блоків буде неможливим. Ця ланцюжкова поведінка також означає, що шифрування блоків потрібно виконувати послідовно, а не паралельно. Однак ці недоліки не поширюються на дешифрування, яке можна виконувати паралельно, якщо всі блоки зашифрованого тексту доступні і можуть допускати втрати блоків.

CTR (скорочення від лічильника) — популярний режим блочного шифрування AES, в якому кожен крок можна виконувати паралельно. CTR подібний до OFB, оскільки він також включає вимкнення послідовності векторів блоків із блоками відкритого та зашифрованого тексту. Основна відмінність полягає в тому, як генеруються ці вектори майданчика.

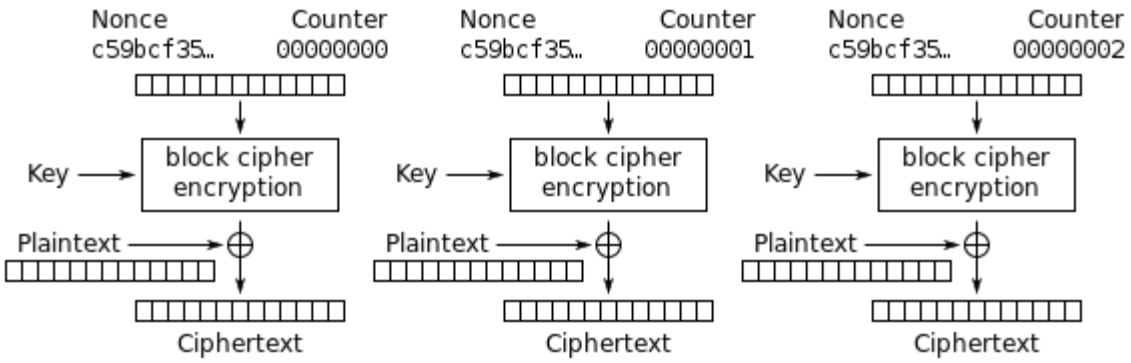


Рисунок 2.9 – Схема шифрування в режимі CTR

У режимі CTR ми починаємо з випадкового насіння, s і обчислюємо вектори pad за формулою:

$$V_i = E_K(s+i-1) \tag{2.5}$$

де E_K позначає алгоритм блочного шифрування з використанням ключа K, V_i - це вектор підкладки, а i - зміщення вектора, починаючи з 1.

Тепер, коли вектори згенеровано, шифрування, подібне до режиму OFB, можна продовжувати за такою формулою:

$$C_i = V_i \oplus B_i \tag{2.6}$$

Розшифровка виконується аналогічним чином:

$$B_i = V_i \oplus C_i \tag{2.7}$$

Оскільки блоки незалежні один від одного в режимі CTR, після того, як вектори блоків були згенеровані, як шифрування, так і дешифрування блоків можна виконувати паралельно. Відсутність взаємозалежності також означає, що режим CTR толерантний до втрати в блоках. Режим CTR вважається дуже безпечним і ефективним для більшості цілей.

Серйозним недоліком CTR є те, що необхідно підтримувати синхронний лічильник як на стороні прийому, так і на кінці. Втрата цього лічильника може призвести до неправильного відновлення відкритого тексту.

Отже, AES реалізовано в безпечних протоколах передачі файлів, таких як FTPS, HTTPS, SFTP, AS2, WebDAVS і OFTP.

Оскільки симетричні та асиметричні алгоритми шифрування мають свої сильні сторони, сучасні безпечні протоколи передачі файлів зазвичай використовують їх комбінацію. Шифри з асиметричним ключем, як і алгоритми шифрування з відкритим ключем, чудово підходять для розподілу ключів і використовуються для шифрування ключа сеансу, який використовується для симетричного шифрування.

Симетричні ключові шифри, такі як AES, більше підходять для шифрування фактичних даних (і команд), оскільки вони вимагають менше ресурсів, а також набагато швидше, ніж асиметричні шифри.

3 ГРАФІЧНИЙ ІНТЕРФЕЙС КОРИСТУВАЧА

3.1 Розробка графічного дизайну симплекс-методу

Сьогодні дизайн задіяний практично у всіх сферах життя суспільства. З кожним днем виробники товарів і послуг все частіше змушені звертатися до майстрів оформлення для того, щоб бути на крок попереду конкурентів, відрізняючись оригінальністю та унікальними рисами, які дозволяють споживачам бачити у вашому бредні щось особливе, притаманне тільки для вас. Прагнення йти в ногу з часом спонукає людей придумувати більш особливі нетривіальні вирішення поставлених задач, тим самим показуючи свою значущість і необхідність потенційному покупцеві.

В розробці кожного додатку графічний інтерфейс відіграє основну роль в сприйнятті користувачем конкретної програми. Проектування візуальної складової є однією з важливих складових прикладного програмного забезпечення. Її мета полягає в тому, щоб покращити ефективність і простоту використання програми. Оскільки в сучасному світі безліч обчислюваних приладів та ще більше програм для них, багато з них потребують певного їх сприйняття з боку користувача. Не двино, що чим кращий інтерфейс, тим ефективніша взаємодія.

На питання яким повинен бути дизайн візуального інтерфейсу складно дати правильну відповідь, адже це досить суб’єктивний погляд. Однак можна виділити такі правила створення хорошого інтерфейсу:

— колір. Кольори поділяються на теплі (як-от жовтий, помаранчевий, червоний), холодні (синій, зелений) та нейтральні (сірий). Зазвичай для створення графічного інтерфейсу використовуються більш теплі кольори, оскільки краще впливають на їх сприйняття користувачем. Варто зазначити, що думка про колір досить суб’єктивна і може змінюватися навіть від настрою користувача;

— форма. В більшості випадків кнопки є прямокутної форми із заокругленими кутами. Кнопки та інші елементи взаємодії повианні реагувати на наведення вказівником мишки та виділятися кольором. Це спрощує сприйняття програми;
— іконки в програмі повинні бути очевидними та відображати їх пряме призначення;
— адаптивний інтерфейс дозволяє користувачеві масштабувати вікно до зручних розмірів без втрати функціональних об’єктів;
— якщо затримки під час виконання якихось обчислень не уникнути, необхідно використовувати індикатор ходу виконання задачі, як-от анімацію.

Графічний інтерфейс користувача — це велика тема, тісно переплетена з психологією, яка займає розуми вчених і сотні сторінок книг і досліджень. В такому малому форматі, що не висловити всю повноту порушеної теми. Однак, дотримання базових принципів дозволить будувати інтерфейси більш доброзичливими до користувача, а так само спростити сам процес проектування.

У дипломній роботі головною метою інтерфейсу є його зручність та його легке візуальне сприйняття. Щоб користуватися програмою не потрібно жодних інструкцій та документацій. На кожному кроці відображаються тільки необхідні для користувача кнопки та поради. Такий підхід дозволяє зосередитися на виконанні завдання, не відволікаючись на інші елементи інтерфейсу.

Розробка графічного інтерфейсу починається з визначення задачі та функціональних потреб, для яких призначений продукт. В основі роботи було зроблено акцент на зручності вводу даних, обчисленнях, які можна проводити за допомогою кліків на потрібні клітинки таблиці в правильній послідовності, в візуальному виділенні допущених помилок, відображення коментарів до кожного кроку розв’язування задачі: текстово та використовуючи gif-анімацію для більш легкого сприйняття інформації.

Взаємодія користувача з програмою починається з головного меню, де можна обрати необхідну задачу: симплекс-метод або транспортна задача. Після вибору однієї із задач, користувач за власним бажанням може повернутися до головного меню програми.

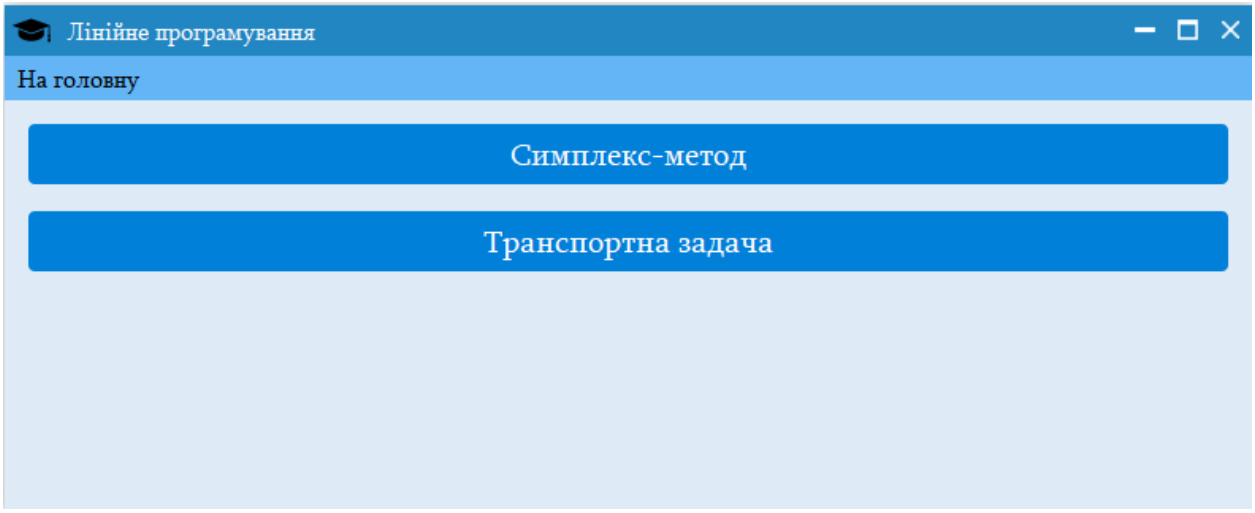


Рисунок 3.1 - Головне вікно програми

Детально розглянемо візуальне представлення розв’язку симплексного методу. Першим етапом є вибір обмежень: необхідно обрати кількість рядків та стовпчиків (кількість змінних). За допомогою стрілок вверх-вниз можна змінювати необхідну кількість відповідних обмежень.

На наступному етапі користувачеві запропоновано ввести обмеження в відповідну таблицю, де кожен рядок — окрема система обмежень, а стовпчик — його аргументи. Знак нерівності можна обрати за допомогою випадаючого списку, після чого справа вказати вільний член. Введення цільової функції відбувається в одному рядку. Екстремум обирається з випадаючого списку та може мати значення min або max. Після натискання на кнопку відбувається перевірка всіх клітинок таблиці і якщо є незаповнені, то такі виділяються червоним кольором, щоб користувач зміг помітити та виправити.

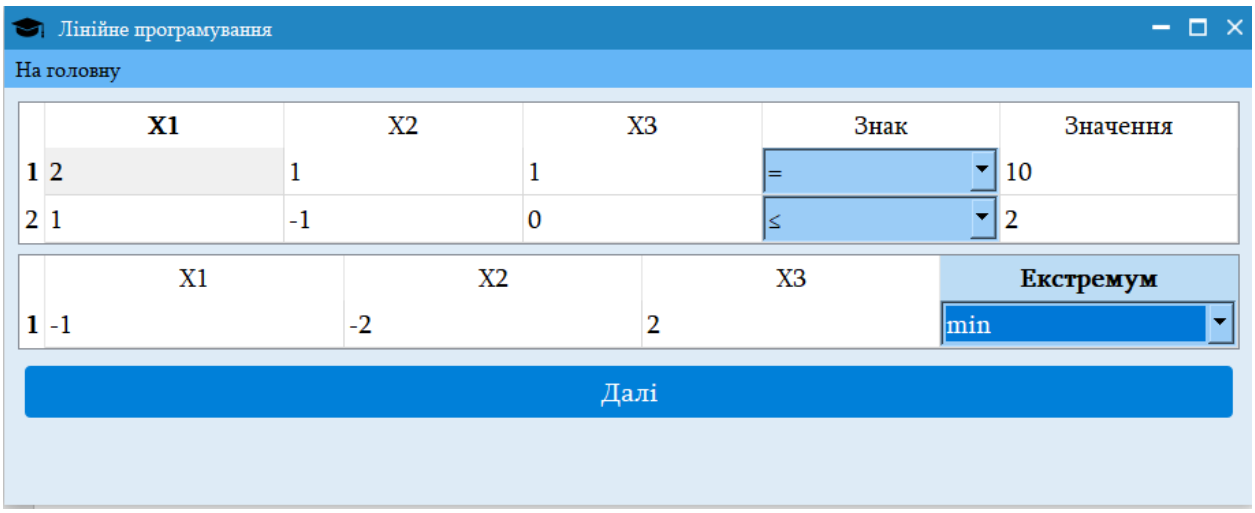


Рисунок 3.2 - Введення обмежень та цільової функції

Після введення всіх обмежень та цільової функції перед користувачем відображаються всі щойно введені дані у текстовому вигляді, що дозволяє ще раз детально перевірити правильність вводу задачі. Далі можна обрати один з методів розв'язування: автоматичний або ручний. За необхідності можна повернутися на крок назад за допомогою відповідної кнопки.

При натисканні на кнопку автоматичного розв'язання перед користувачем відображається отриманий розв'язок введеної задачі. На верхній панелі зліва зображуються обмеження у канонічній формі, а справа — побудована матриця коефіцієнтів. В основній частині вікна (посередині) відображаються пронумеровані симплексні таблиці. Опорний елемент виділяється фіолетовим кольором, а вільні члени — світло-синім.

В нижній частині вікна користувач може переглянути хід розв'язування задачі. Кожен крок коментується програмою, а також вказуються правила, які використовувалися до кожної дії. Наприклад, у коментарях зображається не тільки побудовані канонічні обмеження, а й правила, яких потрібно дотримуватися під час виконання цього етапу завдання. Також отриманий роз'язок з усіма коментарями можна зберегти в PDF-файл для їх зручного перегляду в майбутньому.

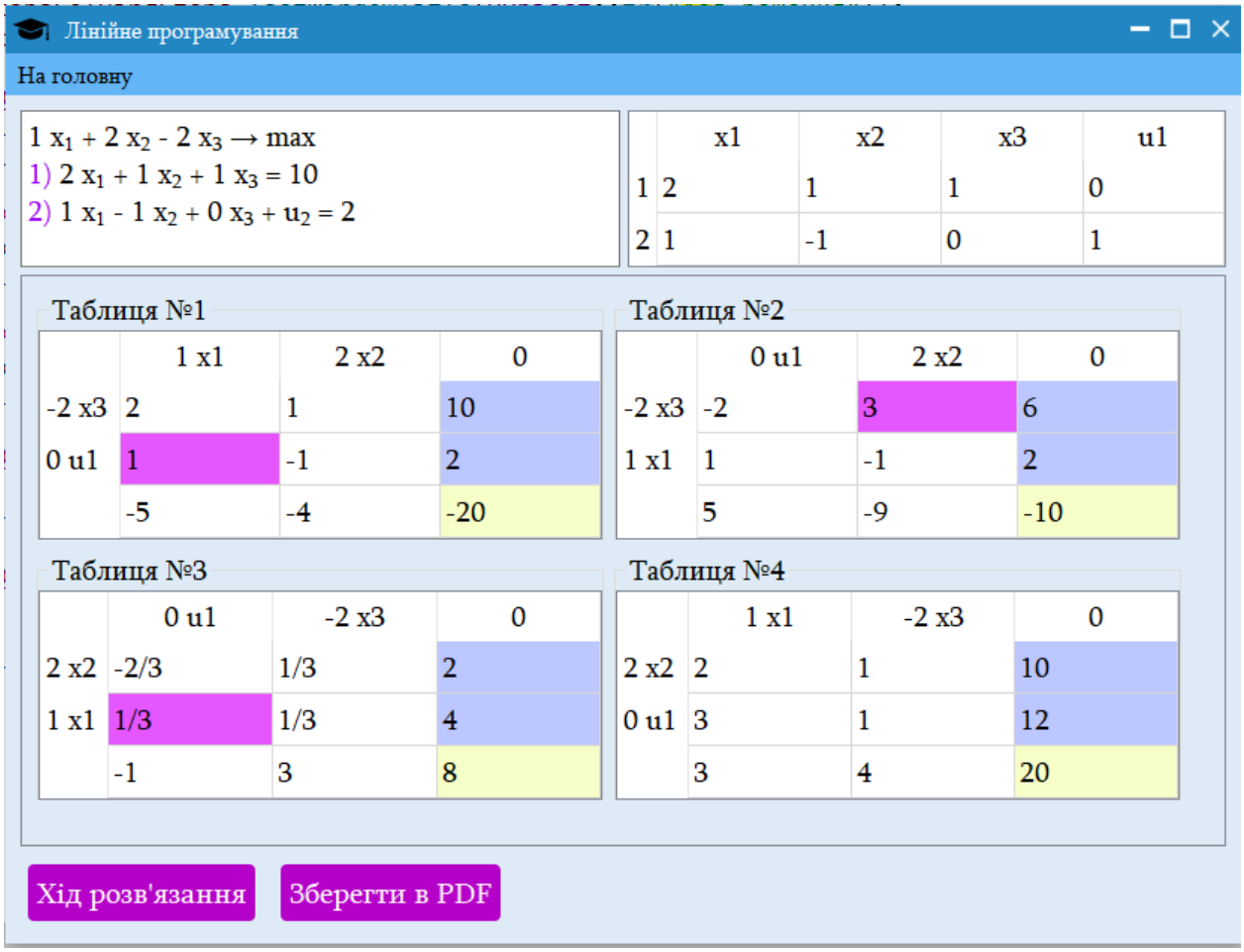


Рисунок 3.3 - Розв’язана задача симплекс-методом

У лівому верхньому куті відображається цільова функція та обмеження у канонічній формі. У правому верхньому куті відобарається побудована матриця коефіцієнтів. Побудовані таблиці відображаються у центрі вікна та нумеруються зліва направо. У кожній таблиці опорний елемент відображається світло-фіотеловим кольором, а вільні члени – голубим.

Внизу вікна відображається функціональна кнопка, натискання на яку відобразить детальні коментарі до поточної розв’язаної задачі. За допомогою кнопки справа у користувача є можливість зберегти отриманий розв’язок із коментарями у PDF-файл.

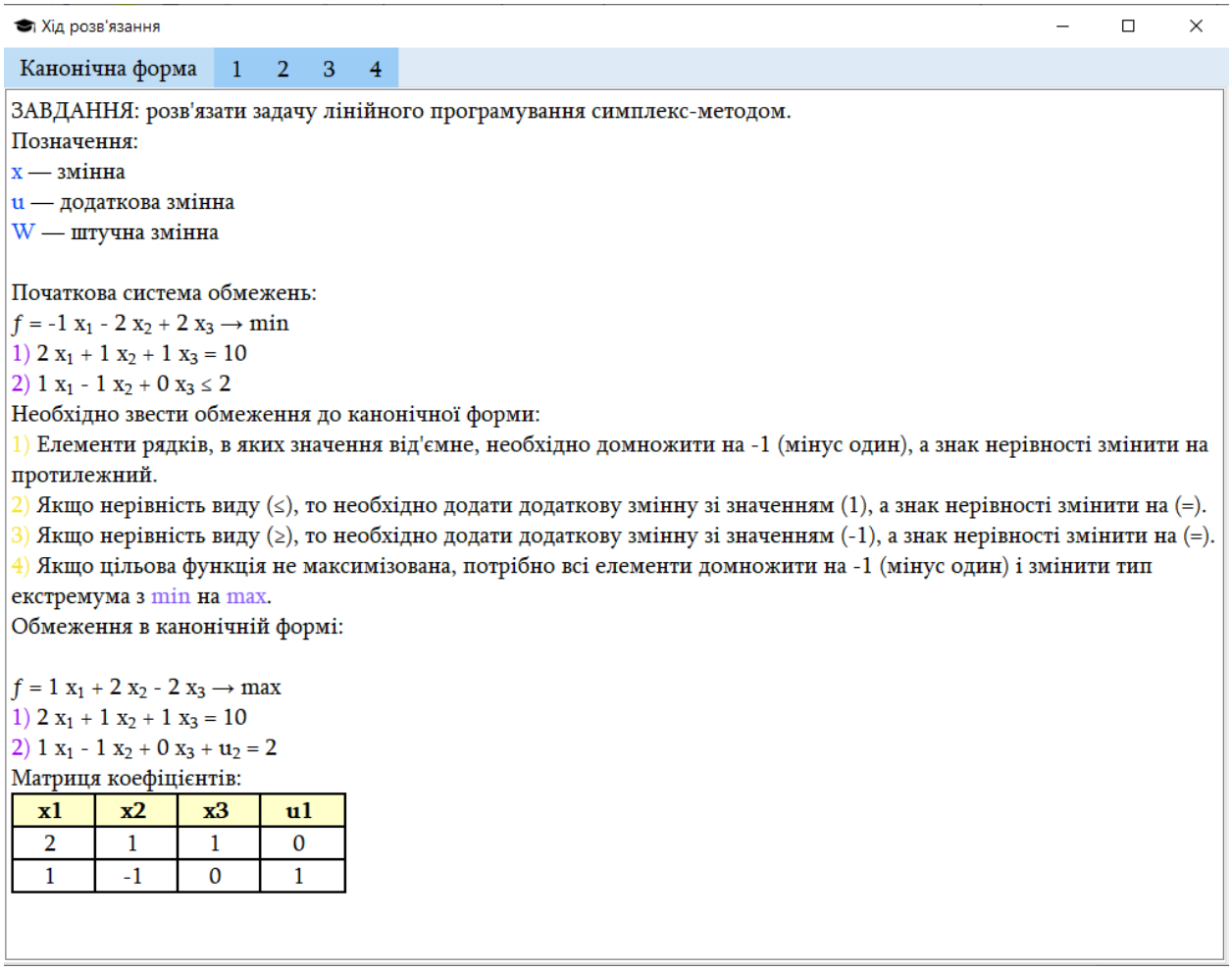


Рисунок 3.4 - Коментарі до розв’язаної задачі

Розв’язання в ручному режимі починається із зведення користувачем системи рівнянь до канонічного вигляду. На цьому етапі дозволяється додати додаткову змінну в систему рівнянь, за допомогою випадаючого списку обрати до кожного рівняння знак рівності, а також за допомогою кнопки змінити знаки аргументів на протилежні. Таким чином, користувачеві не потрібно вручну змінювати кожен знак. Процес максимізації також займає лічені секунди, оскільки зумовлюється тільки вибором екстремума із випадаючого списку. У вікні підказок відображаються поради з необхідних виправлень, які потрібно внести, щоб отримати правильний результат. В нижній частині вікна у користувача є можливість ознайомитися з алгоритмом зведення рівнянь до канонічної форми. Дотримання цих правил значно пришвидшить процес вивчення цього етапу розв’язання поставленої задачі.

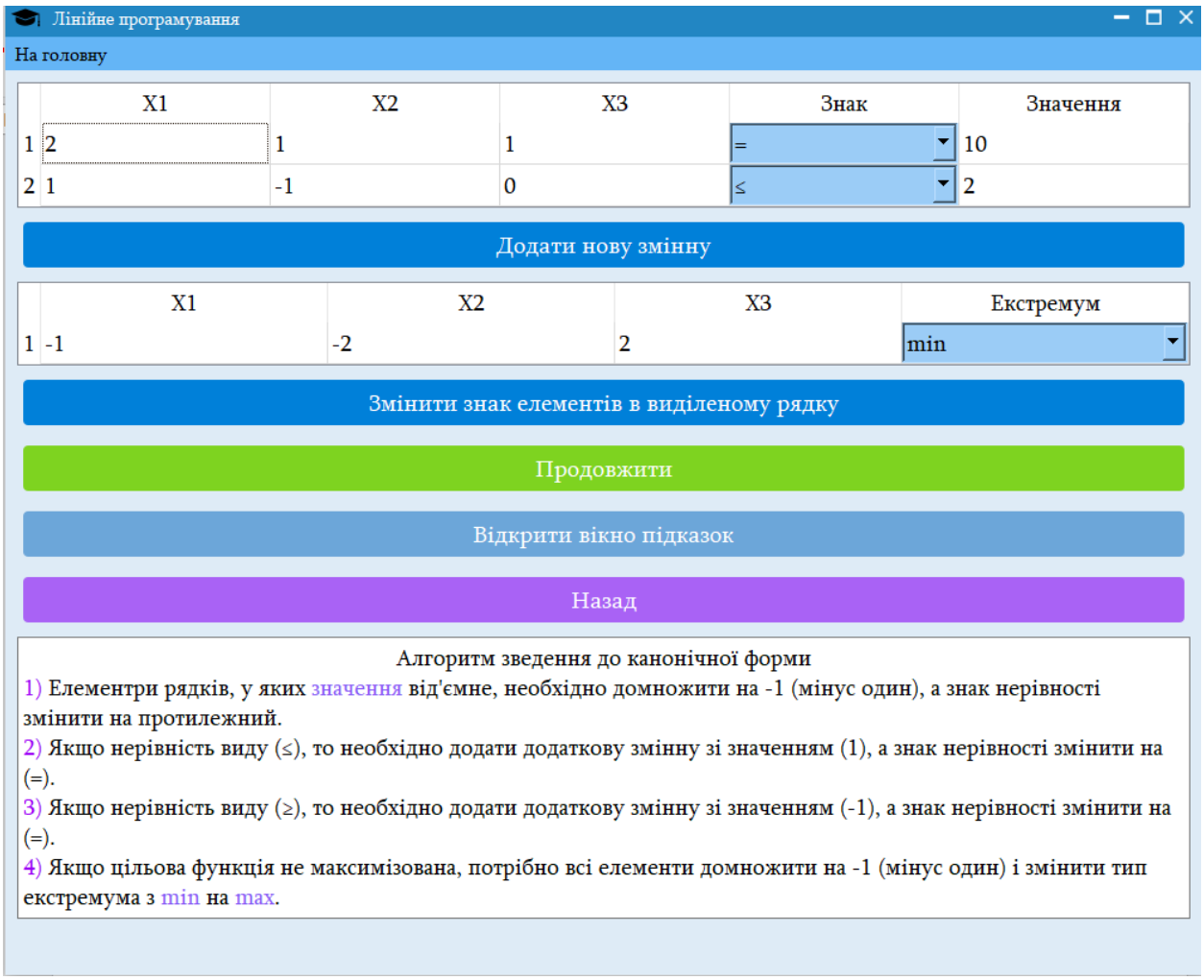


Рисунок 3.5 - Зведення обмежень до канонічної форми

Наступний крок полягає в побудові матриці коефіцієнтів, тобто із системи рівнянь необхідно обрати одиничні стовпчики та, за необхідності, додати їх штучно. Для цього достатньо за допомогою миші клікнути на потрібний рядок та натиснути на кнопку додавання базисного рядка. В нижній частині вікна відображається коментар із поясненням, які стовпчики є базисними та що таке їх повний набір. Пояснення доповнено прикладом, який ілюструє матрицю коефіцієнтів з повним набором базисних стовпчиків.

Після закінчення усіх підготовчих кроків розпочинається безпосереднє ітераційне розв’язування задачі, першим кроком якого є побудова першої симплексної таблиці та заповнення рядка оцінок. Щоб спростити процедуру обчислення, користувачеві достатнього всього лиш клікнути на потрібні клітинки в тому порядку, як би він це виконував під час звичайного

обчислення. Таким чином, враховуючи порядок, програма автоматично вирахує значення конкретної оцінки у відповідному стовпчику. Інформація про те, як це зробити, відображається внизу справа від заповнюваної таблиці у вигляді GIF-анімації. Зліва ж аналогічно відображається поточна послідовність клітинок, на які клікнув користувач.

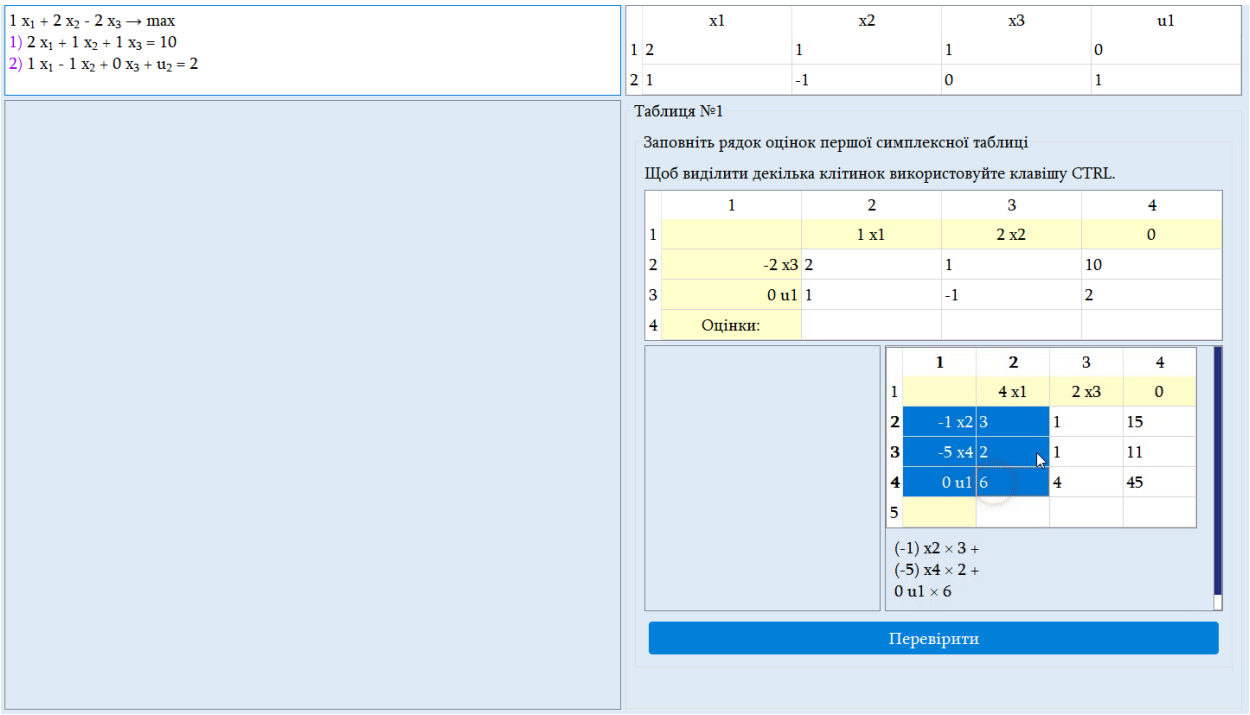


Рисунок 3.6 - Обчислення рядка оцінок першої симплекс-таблиці

Одразу ж після заповнення рядка оцінок користувачеві пропонується перевірити задачу на оптимальність. Коментар до цього кроку внизу вікна дозволяє користувачеві дізнатися, яка задача вважається оптимальною, яка необеженою, а яка розв’язків немає у зв’язку із суперечливими системами обмежень. Якщо ж задача є неоптимальною, користувачеві пропонується перейти до наступного кроку — покращення поточного опорного плану.

Покращення опорного плану починається із вибору опорного рядка та стовпчика. Для цього у додатку є перемикач між рядками та стовпчиками, які візуально їх виділяють. На перетині рядка і стовпчика відображається потенційний опорний елемент. За необхідності, якщо потрібно повернутися до основної задачі (тобто видалити стовпчики зі штучними змінними),

користувач може скористатися функцією видалення виділеного стовпчика. Щоб ознайомитися із правилами пошуку опорного рядка та стовпчика, користувач може скористатися інструкцією, яку можна переглянути, клікнувши на відповідну кнопку.

Останньою ітерацією є перерахування всієї таблиці. Нерідко доводиться витрачати багато часу, щоб уважно обчислити нове значення для кожної клітинки. Ця процедура є досить тривалою, на що людина витрачає досить багато часу. У програмі для того, щоб отримати нове значення в таблиці, достатньо клікнути на потрібні клітинки так, як того вимагає формула. Обчислити, наприклад, значення опорного рядка та стовпчика, можна за допомогою тільки однії кнопки. Таким чином, користувачеві не потрібно витрачати власний час на довготриварі обчислення, що сприятливо вплине на подальший ентузіазм до розв’язування задачі, оскільки всі дії є простими та швидкими. Інструкція для цього етапу, а також поради до заповнення можна переглянути, клікнувши на відповідну кнопку.

3.2 Проектування графічного інтерфейсу транспортної задачі

Розв’язання транспортної задачі розпочинається із введення кількості постачальників та потреб, які є рядками і стовпцями відповідно. За допомогою радіокнопок можна обрати алгоритм розподілу ресурсів: метод “північно-західного кута” або метод “мінімального елемента”. Кориистувачеві запропоновано ввести значення від одного до двадцяти для кількості постачальників та потреб відповідно. Щоб обрати метод розв’язування задачі необхідно клікнути на радіо-кнопку.

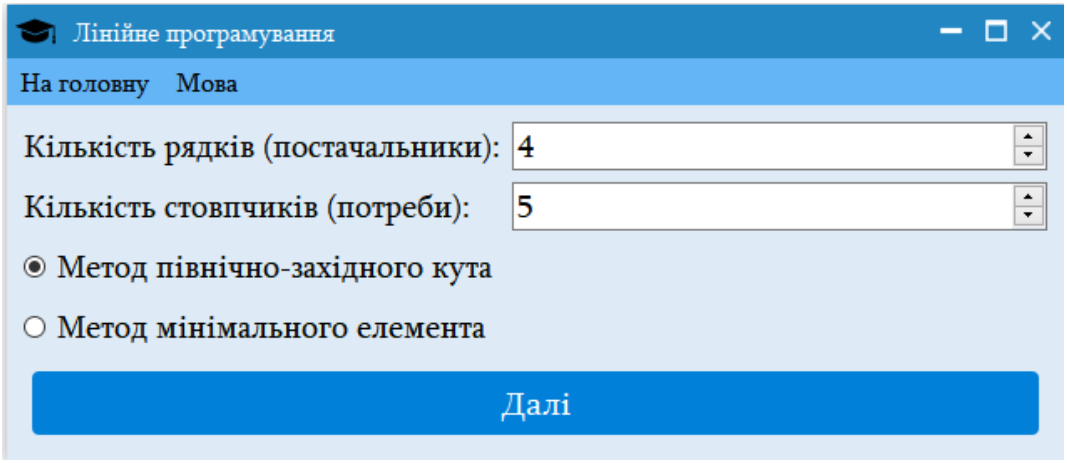


Рисунок 3.7 - Введення кількості обмежень та алгоритм розподілу ресурсів транспортної задачі

На наступному кроці необхідно ввести матрицю тарифів, значення запасів та потреб. Рядок та стовпчик для цих величин виділяється кольором.

	B1	B2	B3	B4	B5	**Запаси**
A1	2	3	9	7	2	37
A2	3	4	6	1	5	16
A3	5	1	2	2	1	14
A4	4	5	8	1	9	11
Потреби	16	18	12	15	17	

Рисунок 3.8 - Введення матриці тарифів транспортної задачі

Після введення задачі користувачеві пропонується обрати спосіб розв’язування задачі: автоматично або вручну — в навчальному режимі.

Кінцевим елементом графічного інтерфейсу є відображення всіх побудованих таблиць до задачі. Розглянувши таблицю розподілу ресурсів

(див. рис.), варто зазначити, що для більш якісного її сприйняття відображаються кроки побудови цієї таблиці у квадратних дужках, де спочатку вказується номер ітерації, а далі, через кому, напрямок викреслювання: рядок або стовпчик.

У подальших таблицях для зручності елементи таблиці виділяються кольором. Так, потенціали відображаються фіолетовим кольором, значення перевезення зеленим кольором.

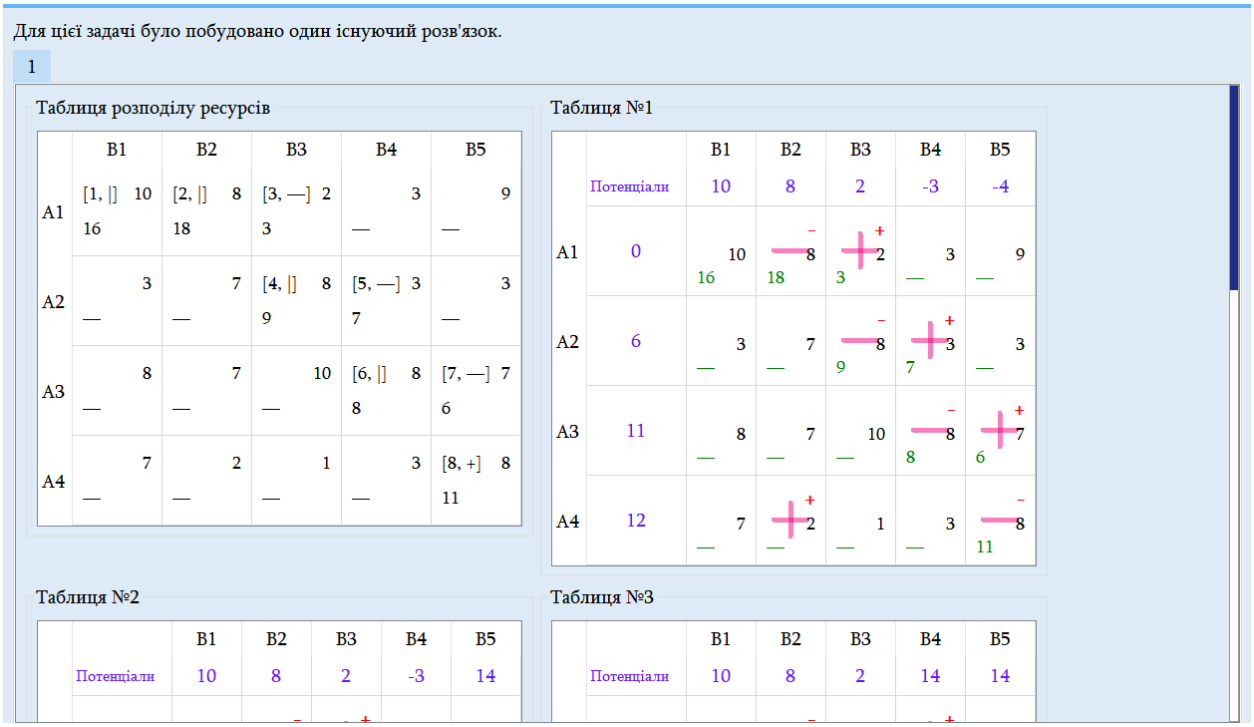


Рисунок 3.9 - Транспортна задача, розв’язана методом “північно-західного кута”

Розв’язування у навчальному режимі починається зі зведення задачі до закритого типу. Користувачеві пропонується переглянути таблицю та за необхідності додати додатковий рядок або стовпчик за допомогою відповідних функціональних кнопок. Внизу вікна відображається коротка інструкція із поясненням, яка задача називається відкритою та закритою. Навіть якщо користувач вперше розв’язує транспортну задачу, ознайомившись з інструкцією він зможе зробити правильний вибір.

Рисунок 3.10 - Зведення транспортної задачі до закритого типу

Звівши задачу до закритого типу користувачеві пропонується обрати метод розподілу ресурсів. У програмі їх є два: звичайний та спрощений. Звичайний полягає в тому, що користувач нічим не обмежений під час розподілу ресурсів: він може обирати будь-яку клітинку, вказувати довільне значення перевезення та викреслювати будь-який рядок або стовпчик. Очікується, що користувач, який обрав цей метод, вже має певне поняття правил розподілу ресурсів. У корисувача є можливість в будь-який момент перевірити правильність своїх дій та у випадку помилки програма вкаже на клітинку, яка не збігається з очікуваним результатом.

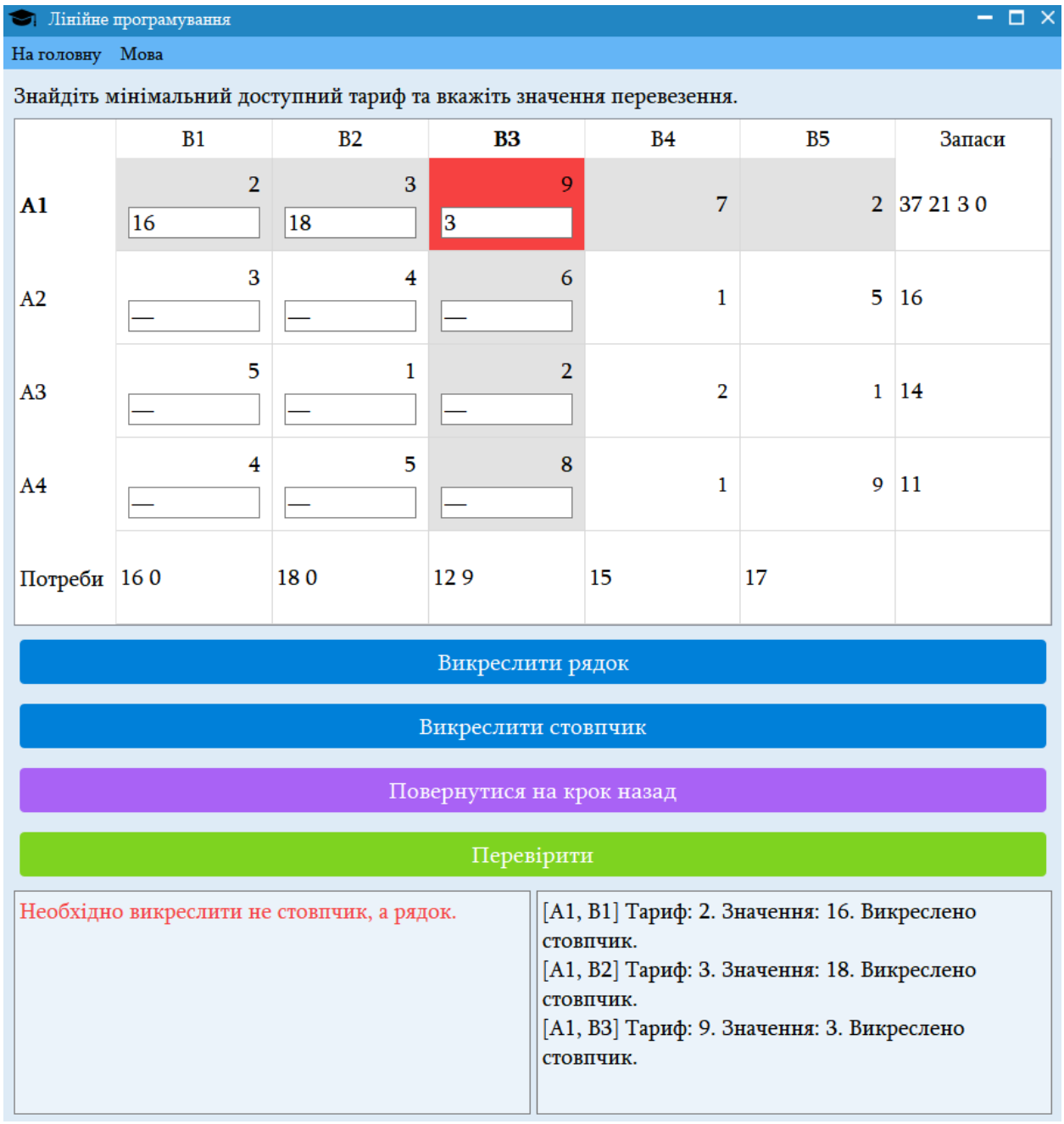


Рисунок 3.11 - Розподіл ресурсів транспортної задачі звичайним методом

Спрощеший метод розподілу ресурсів дозволяє користувачеві ознайомитися з алгоритмом поставної задачі. У цьому випадку користувач може обирати клітинки, які виділяються кольором, тобто які є доступними. Обравши клітинку дозволяється обрати значення перевезення із запропонованих у випадаючому списку значень, а далі викреслити необхідний рядок або стовпчик. Оскільки обрано спрощений режим, під час першої ж

помилки буде зображено підказку. Таким чином, користувач не зможе жодним чином допуститися помилки, оскільки це неможливо на програмному рівні.

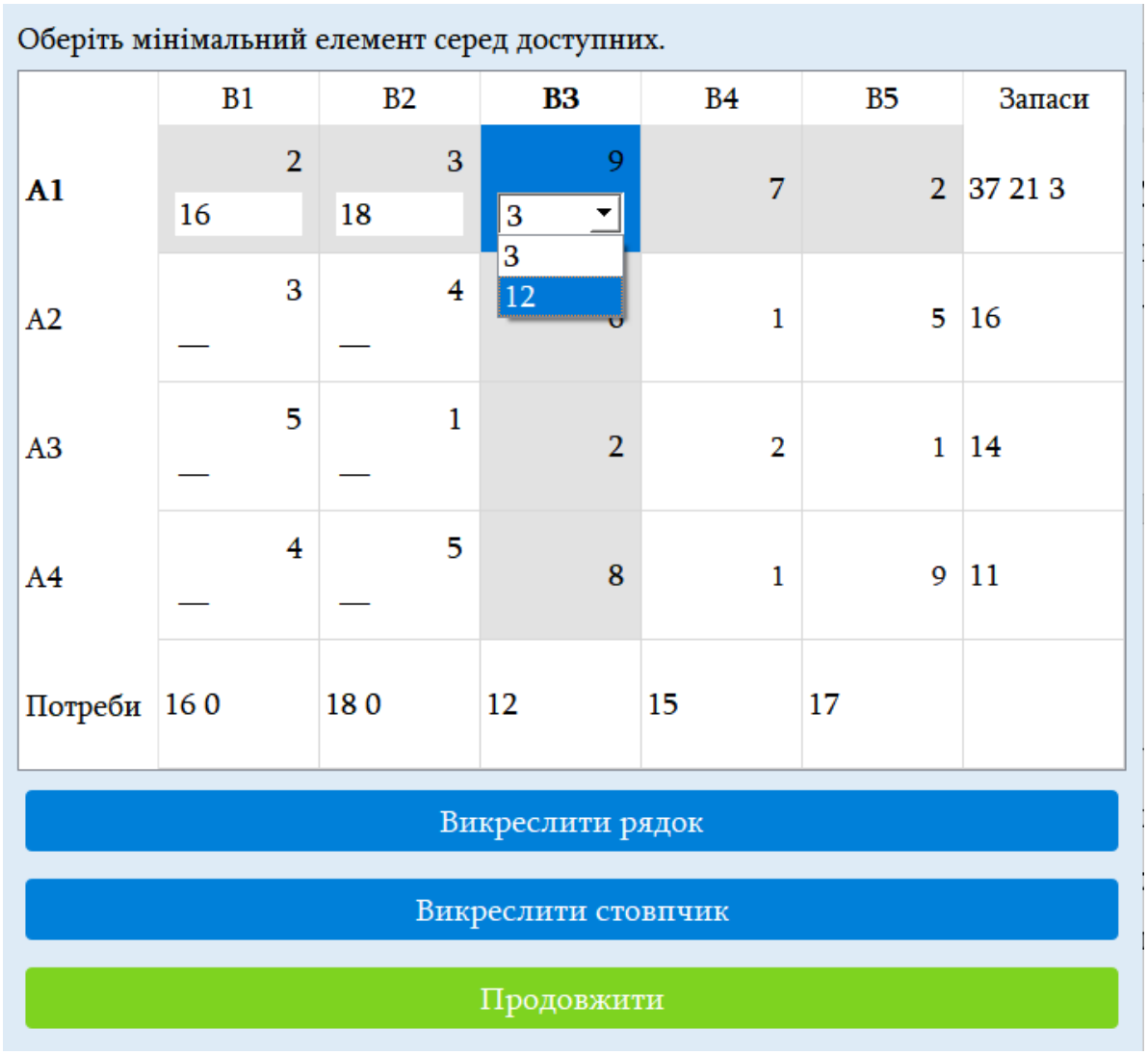


Рисунок 3.12 - Розподіл ресурсів транспортної задачі спрощеним методом

Наступним кроком розв’язування є побудова потенціалів рядка і стовпця. Для кращого розуміння цей крок також відображає інструкцію внизу вікна із поняттям потенціала та алгоритмом його побудови.

Після побудови потенціалів необхідно обрати неоптимальну клітинку, тобто ту клітинку, де не виконується умова, за якої сума значень конкретних потенціалів рядка і стовпчика повинна бути меншою від значення тарифу. Обравши неоптимальний стовпчик, користувачеві запропоновано побудувати

цикл, тобто відмітити необхідні клітинки знаком “+” або “-”. Інструкція щодо побудови циклу відображаться внизу вікна.

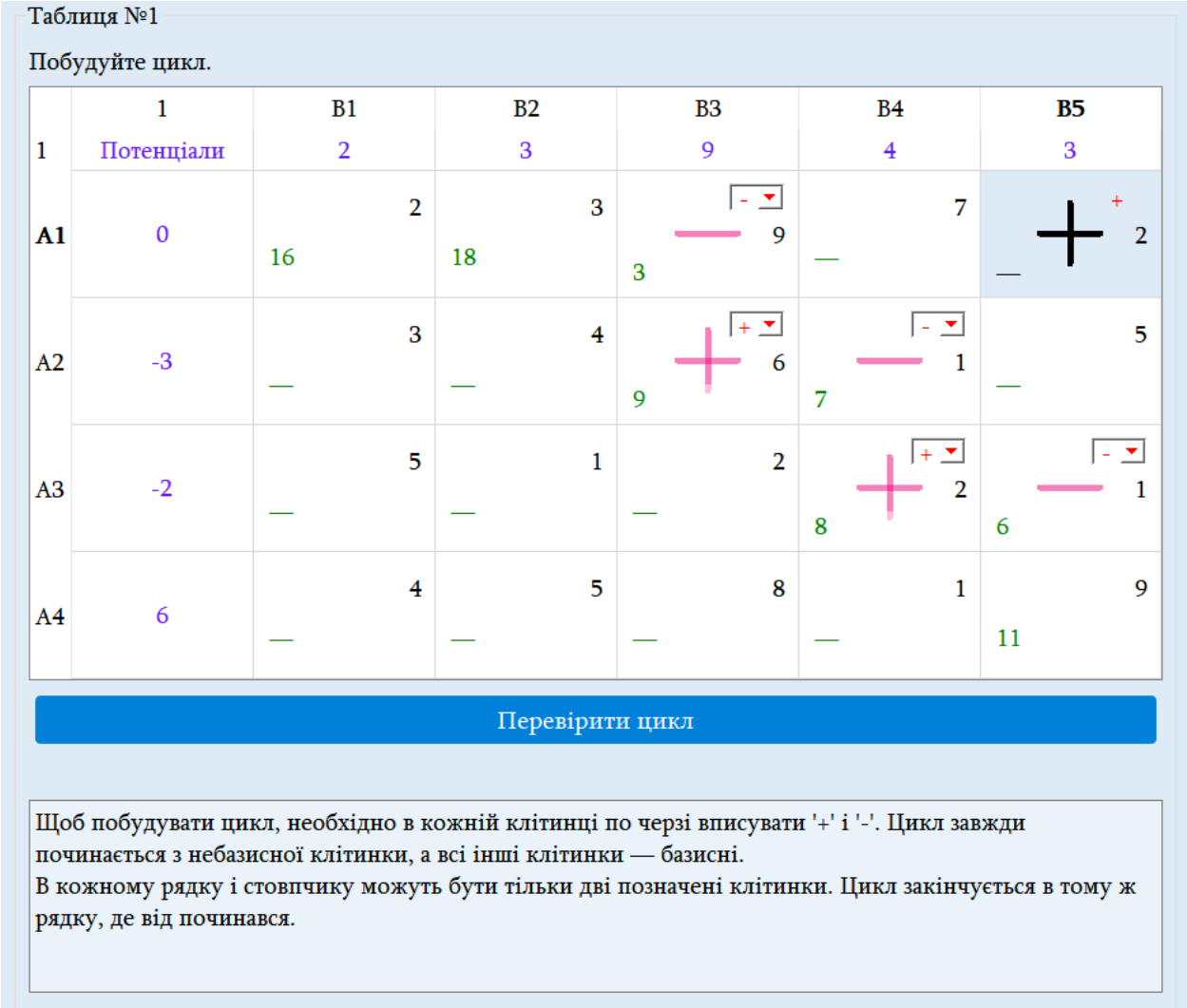


Рисунок 3.13 - Побудова циклу до транспортної задачі

Останнім кроком є пошук мінімального значення у клітинках зі знаком “-”, та внесення відповідних змін до таблиці: у клітинки зі знаком “+” необхідно до поточного значення додати щойно знайдене мінімальне значення, а у клітинки зі знаком “-” від поточного значення необхідно відняти отримане мінімальне значення.

Ввівши дані правильно, користувачеві буде запропоновано знову заповнити рядок та стовпчик потенціалів, тобто повторити всі попередні дії або закінчити розв’язування задачі, якщо план виявиться оптимальним.

3.3 Інформаційна система обліку успішності студентів

У користувачів є можливість не тільки навчитися розв’язувати задачі лінійного програмування, а й переглянути свій результат після вирішення задачі. Якість одержаних знань характеризує ефективність спільної навчальної роботи професорсько-викладацького складу і студентів. Об’єктивне уявлення про якість знань студентів можна одержати тільки при систематичному контролі навчальних досягнень студентів. Результати навчальної роботи та якість знань студентів, виражаються в оцінках. Оцінка - це визначення ступеня засвоєння студентами знань, умінь і навичок відповідно до вимог програм та керівних документів, по яких здійснюється навчання. При перевірці знань, умінь і навичок велике значення має їх об'єктивна та незаангажована оцінка до якої висуваються наступні вимоги: оцінка повинна бути об'єктивною і справедливою, ясною і зрозумілою для студента, оцінка повинна виконувати стимулюючу функцію та орієнтувати на навчання. Після розв’язання задачі оцінку можна зберегти та відправити викладачеві. Результат, який зберігає студент, шифрується за допомогою алгоритму AES, що гарантує те, що дані не буде відредаговано після їх збереження. Ключ розшифрування зберігається тільки на комп’ютері викладача.

Особливістю такої системи є навіть те, що робота, яку зберігає користувач, підписана цифровим відбитком пристрою — інформацією, яка зібрана про пристрій для подальшої ідентифікації. Таким чином виконання декількох робіт на одному присторії із різним ім’ям та прізвищем, недопустимі, оскільки на комп’ютері ПК програма за допомогою цифрового відбитку ПК одразу визначить, кому належить виконана робота.

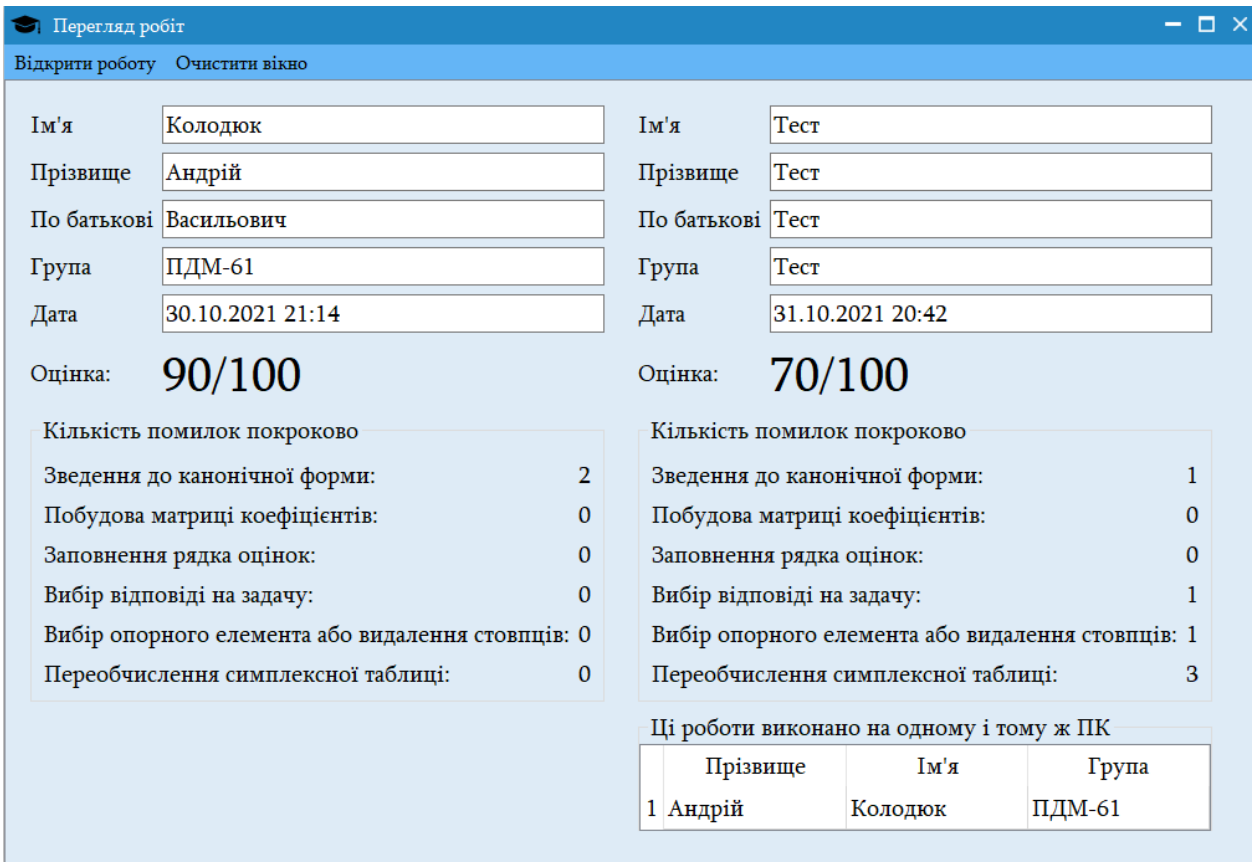


Рисунок 3.14 – Перегляд результатів розв’язування задач

ВИСНОВОК

На основі проведеного порівняльного аналізу вже існуючих прогамних забезпечень для розв’язування задач лінійного програмування було створено вузьконаправлене програмне забезпечення для розв’язування “Симплекс-методу” і “Транспортної задачі”. Так, у розробленому додатку додано переваги із усіх досліджених програм. Враховуючи основний недолік розглянутих програмних забезпечень, який полягає в складності взаємодії користувача із програмою, великий акцент зроблено на те, щоб користувачі без спеціальних знань та жодної підготовки змогли розпочати розв’язування задач. Перевагою розробленого додатку є можливість покрокового розв’язку обраної задачі, де кожен крок користувача перевіряється комп’ютером, а також надаються короткі теоретичні відомості та рекомендації для правильного розв’язування задачі на кожному кроці.

Отже, ознайомившись із існуючими рішеннями програмних забезпечень було розроблено додаток, який є конкурентно-спроможним та володіє певними особливостями, такими як автоматичний та навчальний розв’язок задач із коментарями до них, що дозволяє користувачам не тільки отримати відповідь на конкретну задачу, а й зрозуміти хід розв’язування задачі. Таким чином, розроблене програмне забезпечення дозволить значно пришвидшити швидкість навчання майбутнів фахівців, а також зробить цей процес цікавішим.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Черняк А.А., Черняк Ж.А., Доманова Ю.А. «Вища математика на базі MathCad» / Київ, КОРОНА, 2004. -150 с.
2. Крушевський А.В. Математичне програмування в економіці та управлінні : навч.-метод. посіб. / Київ, КНЕУ, 2001. -107 с.
3. Ільченко М.Ю., Кравчук С.О. Сучасні телекомунікаційні системи / Київ, НВП «Видавництво «Наукова думка» НАН України», 2008. -328 с.
4. Кудрявцев Е.М. MathCad 2000: розв'язування задач / Харків, Вища школа, 2001. -34 с.
5. Лунгу К. Н. Лінійне програмування. Посібник для розв'язування задач / Харків, ХДТУБА, 2005. -99 с.
6. Гончаренко Я.В. Математичне програмування / Київ, НПУ імені М.П. Драгоманова, 2010. -184 с.
7. Белобродський А.В., Гриценко М.А. Пошук рішень з Excel / Київ, ІВЦ «Видавництво «Політехніка», 2011. -136 с.
8. Катренко А.В. Дослідження операцій : навч. посіб. / Львів, Магнолія Плюс, 2004. -549 с.
9. Наконечний С.І. Математичне програмування : навч. посіб. / Київ, КНЕУ, 2003. -452 с.
10. Самойленко М.І. Математичне програмування. / Харків, Основа, 2002. -424 с.
11. Зайченко О.Ю., Зайченко Ю. П. Дослідження операцій. Збірник задач / Київ, Видавничий Дім «Слово», 2007. -472 с.
12. Ларіонов Ю.І., Левикін В.М., Хажмурадов М.А. Дослідження операцій в інформаційних системах / Харків, СМІТ, 2005. -364 с.
13. Лотов А.В. Вступ в математичне моделювання. / Харків, ХДАМГ, 1997. -103 с.
14. Мазаракі А.А. Математичне програмування в Excel : навч. посібник для студ. екон. спец. внз. / Київ, Четверта хвиля, 1998. – 207 с.

15. Самсонов В.В. Алгоритми розв’язання задач оптимізації : навчальний посібник / Київ, НУХТ, 2014. -300 с.

ДОДАТКИ

Додаток А

Таблиця 1. Порівняння вимог до програмного забезпечення, переваги та недоліки програм математичного програмування

	Мінімальні вимоги до системи	**Переваги**	**Недоліки**
Excel	створені різні варіанти програмного забезпечення для відповідних операційних систем	Порівняно дешевий пакет (існують безкоштовні версії)	складності одночасної роботи декількох користувачів з однією інформаційною системою
		простота освоєння	підвищений вплив «людського фактора» на коректність таблиць
		гнучкість	складності організації безлічі таблиць в різних варіантах (чорновий, узгоджений, затверджений і т.п.), що вимагає додаткових зусиль по організації роботи з файлами
		потужні інструменти обробки даних (включаючи статистичні функції) і презентацій результатів	складність із забезпеченням конфіденційності даних інформаційної системи. [4].
		практично необмежені можливості з обміну даними з іншими інформаційними системами	низька продуктивність

Продовження таблиці 1.

		порівняно низькі витрати на впровадження системи бюджетування і на її модернізацію	
MathLab	процесор Pentium III, 4, Xeon, Pentium M; AMD Athlon, Athlon XP, Athlon MP	потужність	Невисока інтегрованість середовища
	256 Мбайт оперативної пам'яті (рекомендується 512 Мбайт)	надійність	не надто виразна довідкова система
	400 Мбайт дискового простору (тільки для самої системи MatLab і її Help) [3]	прозорість	
	операційна система Microsoft Windows 2000, (SP3) / XP	універсальність	специфічний редактор коду MatLab-програм
		гнучкість	
		Розширення можливостей досягається за рахунок використання великої кількості спеціально розроблених пакетів розширення, наборів інструментів. [6].	
MathCad	процесор Pentium II або вище	природна математична мова	обмежені можливості існуючих операторів
	128 Мбайт оперативної пам'яті (рекомендується 256 Мбайт або більше);	Наочність	труднощі реалізації складних алгоритмів
	200-400 Мбайт дискового простору	Гарна діагностика помилок	
	операційні системи: Windows 98 / Me / NT 4.0 / 2000 / XP	Висока точність обчислень	
		Забезпечення роботи з комплексними числами	

Продовження таблиці 1.

		Реалізація багатьох стандартних функцій обчислювальної математики	
		Можливості символьних математичних перетворень	
Mathematica	процесор Pentium II або вище	вирішення більшості математичних задач в діалоговому режимі без традиційного програмування	незвичайна мова програмування
	128 Мбайт оперативної пам`яті (рекомендується 256 Мбайт або більше)	сильна і вишукана графіка	
	400-550 Мбайт дискового простору		
	операційні системи: Windows 98/Me/ NT 4.0/2000/2003 Server/2003x64/XP/XP x64.		
Maple	процесор Pentium III 650 МГц	здатність виконувати арифметичні дії в символьному вигляді	Дорога вартість
	128 Мбайт оперативної пам'яті (рекомендується 256 Мбайт)	При роботі з дробами і корінням програма виробляє необхідні скорочення і перетворення в стовпчик, що дозволяє уникнути помилок при округленні	Задумливість
	400 Мбайт дискового простору		
	операційні системи: Windows NT 4 (SP5) / 98 / ME / 2000/2003 Server / XP Pro / XP Home		

Додаток Б

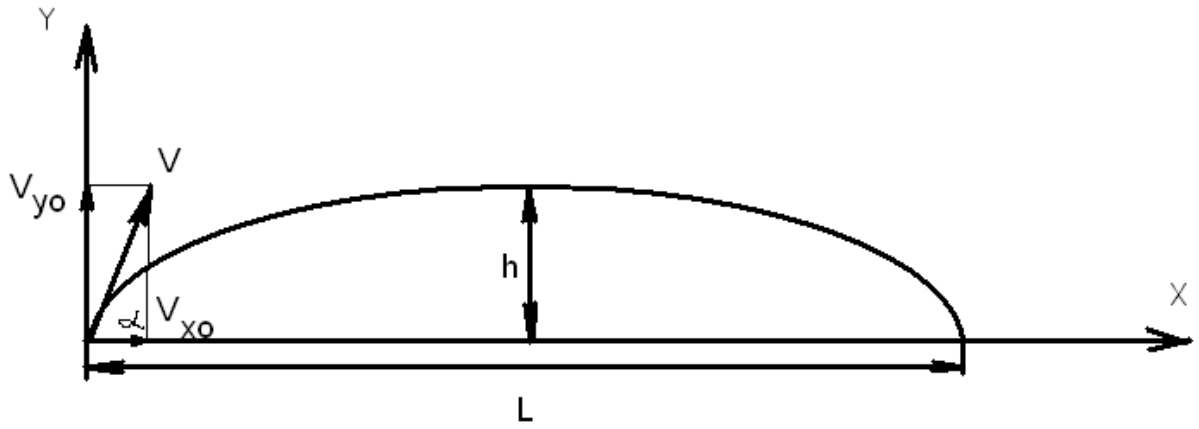
Таблиця 2. Порівняння можливостей програм математичного моделювання

Критерій порівняння	Excel	MathLab	MathTab	Mathematica	Maple
Зручність інтерфейсу та встановлення	Легко налаштовується	Достатньо складне налаштування для звичайного користувача	Достатньо складне налаштування для звичайного користувача	Достатньо складне налаштування для звичайного користувача	Достатньо складне налаштування для звичайного користувача
Мова інтерфейсу	Українська	Англійська	Англійська	Англійська	Англійська
Мова довідки	Українська	Англійська	Англійська	Англійська	Англійська
Підтримка локальної мережі та Інтернету	ЛМ та відправка збережених даних на e-mail	ЛМ, Інтернет	ЛМ, Інтернет	ЛМ, Інтернет	ЛМ, Інтернет
Збереження результатів	+	+	+	+	+

Додаток В

Розв’язування завдання за допомогою програм Mathcad та MatLab

Завдання**.** Знайти вид залежності горизонтальної довжини польоту тіла і максимальної висоти траєкторії від одного з коефіцієнтів опору середовища, фіксувати всі інші параметри. Уявити цю залежність графічно і підібрати аналітичну формулу.



Текст програми в середовищі MathCad

$x(t) = V\mathrm{x}_0 t$

$y(t) = Vy_0 t - \frac{gt^2}{2}$

$y(tl) = Vy_0 tl - \frac{gtl^2}{2} = 0$

$Vx_0 = V\cos\alpha$

$Vy_0 = V\sin\alpha$

$$tl = \frac{2Vy_0}{g}$$

$$tl = \frac{2V \sin\alpha}{g}$$

Час польоту до найвищої точки траєкторії дорівнює половині часу t1, тому висота траєкторії дорівнює:

$y(t) = Vy_0 t - \frac{gt^2}{2}$.

$H = y\left(\frac{tl}{2}\right) = \frac{Vy_0{}^2}{g} - \frac{g}{2} * \frac{Vy_0{}^2}{g^2} = \frac{V^2 * ((\sin\alpha)^2)}{2g}$.

Дальність польоту в горизонтальному напрямку:

$\mu = \frac{\cos\alpha}{\sin\alpha}$ – коефіцієнт супротиву середовища.

$L = x(tl) = \frac{Vx_0 2Vy_0}{g} = \frac{2V^2 \sin\alpha * \cos\alpha}{g}$,

$(\sin\alpha)^2 = \frac{1}{1+(\cot\alpha)^2)} = \frac{1}{1+\mu^2}$,

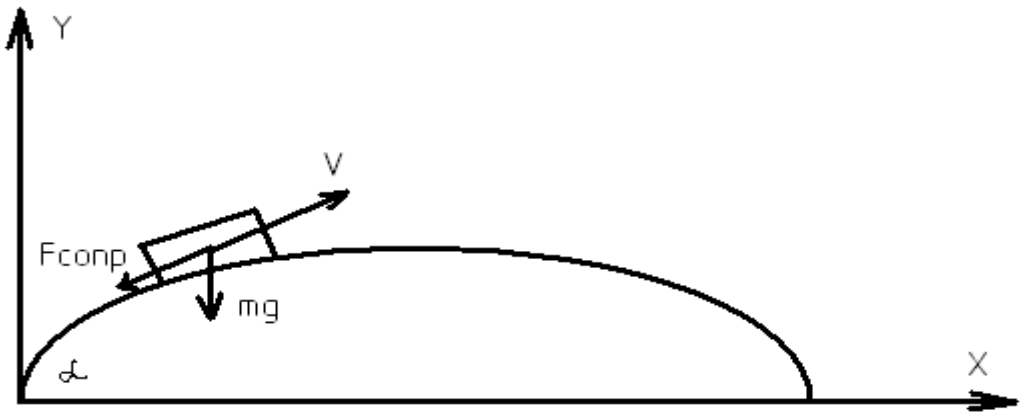
$(\cos\alpha)^2 = \frac{1}{1+(\tan\alpha)^2} = \frac{\mu^2}{1+\mu^2}$.

Т.ч. отримаємо такі формули:

$H(\mu) = \frac{V^2}{2g} * \frac{1}{1+\mu^2}$,

$L(\mu) = \frac{2V^2}{g} * \frac{\mu}{1+\mu^2}$.

Побудуємо графіки залежностей максимальної висоти і довжини польоту тіла від коефіцієнта опору середовища:



$V := 10$ – швидкість тіла

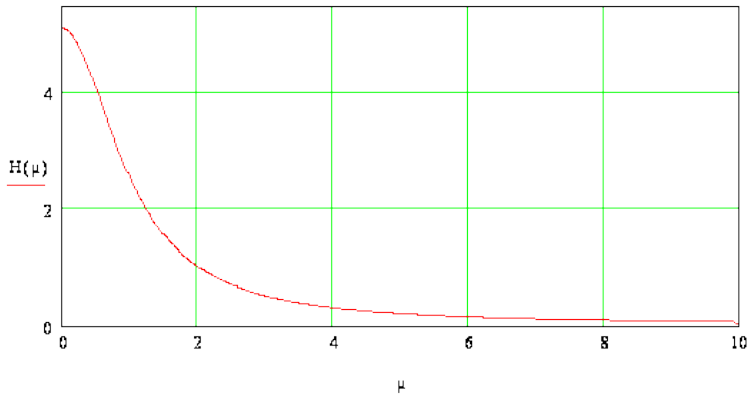
$g := 9.8$ – прискорення вільного падіння

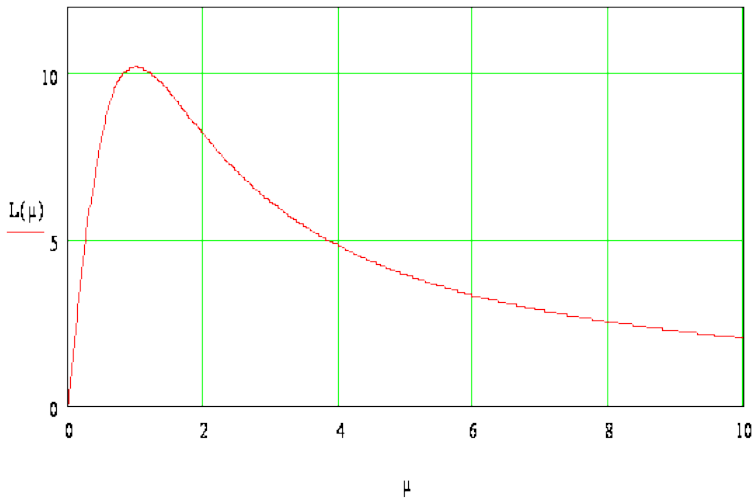
$$H(\mu) := \frac{V^2}{2g} * \frac{1}{1+\mu^2}$$

H - залежність висоти польоту тіла від коефіцієнта опору середовища

$$L(\mu) := \frac{2V^2}{g} * \frac{1}{1+\mu^2}$$

L - залежність горизонтальної довжини польоту тіла від коефіцієнта опору середовища



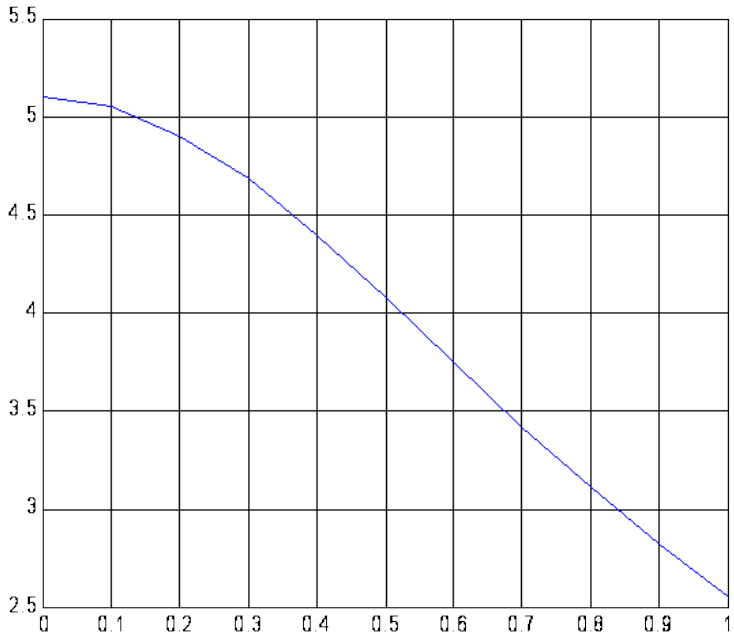


Тексти програм в середовищі Matlab

Частина 1:

```
grid on;
hold on;
g=9.81;
V=10;
m=0:0.1:1
H=(V^2)./(2*g*(1+m.^2));
plot(m,H);
```

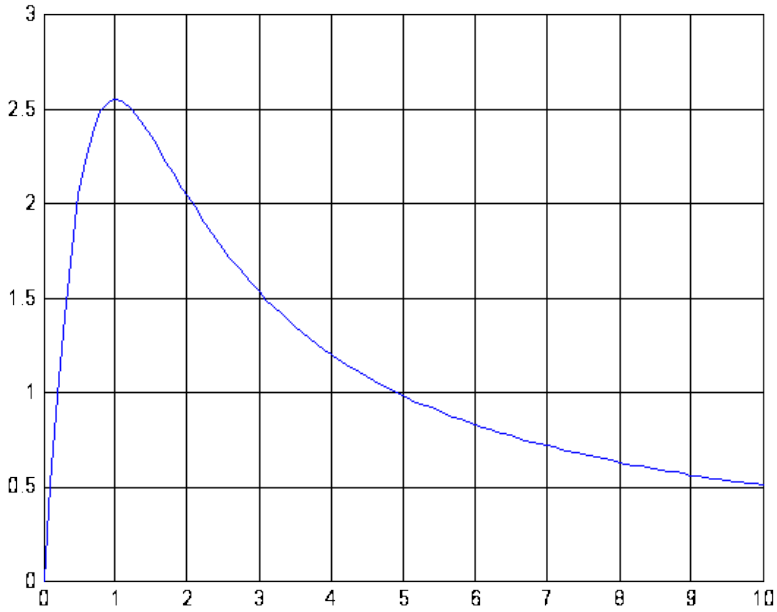
Графік:



Частина 2:

```
grid on;
hold on;
g=9.81;
V=10;
m=0:0.1:1
L=(m*(V^2))./(2*g*(1+m.^2));
plot(m,L);
```

Графік:

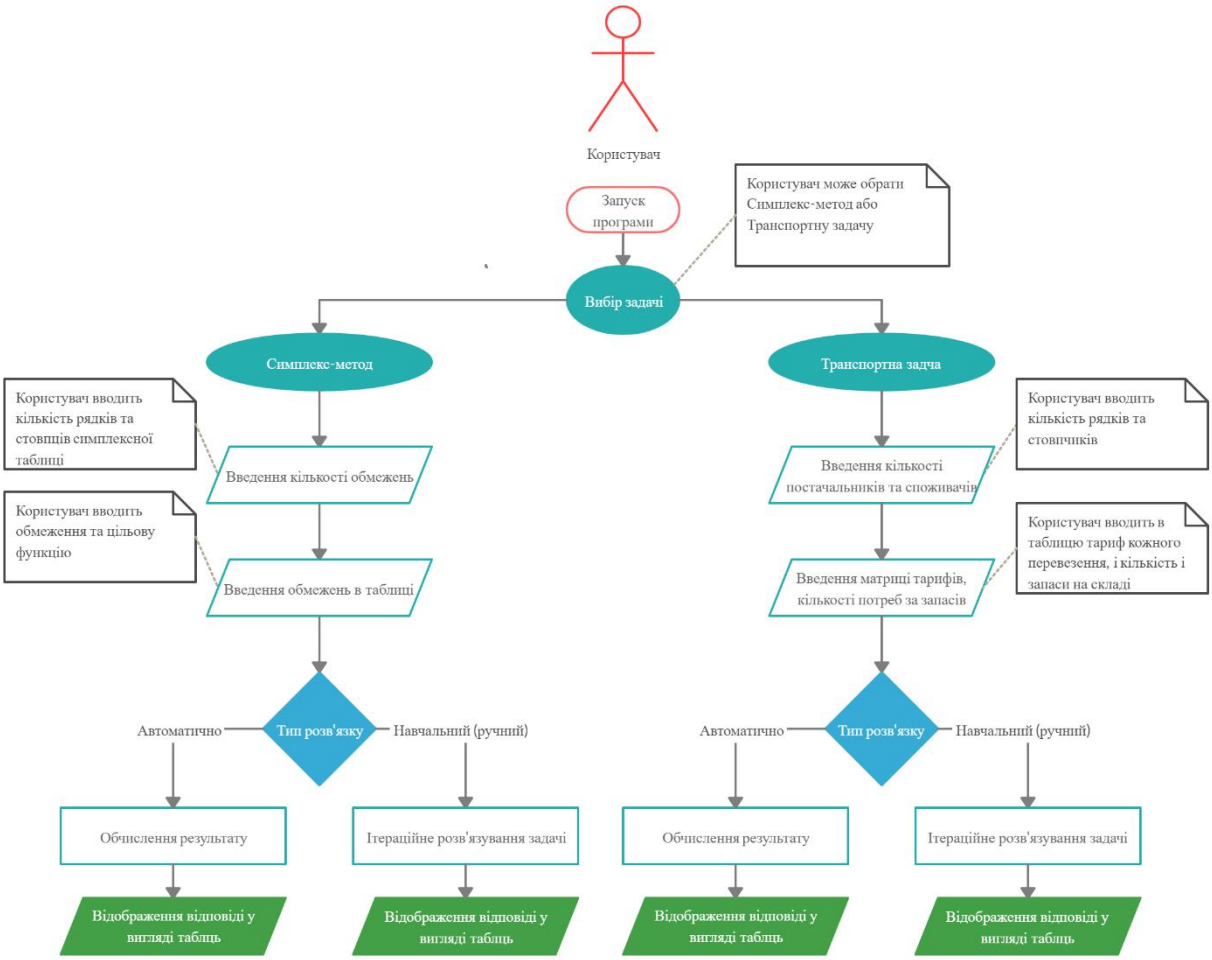


Висновок

З графіків видно, що висота польоту тіла, як і довжина польоту тіла зменшується при збільшенні опору середовища (повітря). Графіки, побудовані в різних математичних програмах збігаються, отже, можна зробити висновок, що математичне моделювання зроблено правильно.

Додаток Г

Діаграма взаємодії користувача з програмою



Додаток Д

Презентація

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО–НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра інженерії програмного забезпечення

Магістерська робота

«Удосконалення інформаційної технології пошуку оптимального рішення на основі методів оптимізації»

Виконав: Колодюк Андрій Васильович
Керівник: Жебка Вікторія Вікторівна

Київ - 2021

МЕТА, ОБ'ЄКТА ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ 2

Мета роботи: підвищення ефективності визначення оптимального рішення за допомогою інформаційної технології розробленої на основі методів оптимізації

Об'єкт дослідження: пошук оптимального рішення на основі методів оптимізації

Предмет дослідження: інформаційна технологія розв'язування задач лінійного програмування

АКТУАЛЬНІСТЬ РОБОТИ

Недоліки існуючих розробок:

- Направленість на розв'язання задачі (отримання конкретної відповіді)
- Направленість на розв'язання задачі (отримання конкретної відповіді)
- Відсутність формування детального звіту кроків розв'язування задачі

Переваги запропонованого ПО:

- Можливість покрокового розв'язання всіх етапів із перевіркою комп'ютера
- Формування детального звіту для усіх етапів розв'язання у PDF
- Можливість оцінювання та перегляду оцінок на ПК викладача

3

Порівняння вимог до програмного забезпечення, переваги та недоліки програм математичного програмування

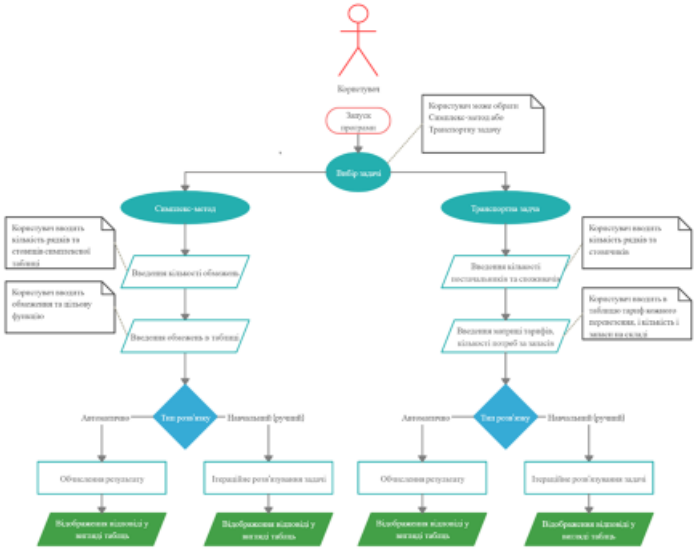
	Мінімальні вимоги до системи	Переваги	Недоліки
Excel	створені різні варіанти програмного забезпечення для відповідних операційних систем	Порівняно дешевий пакет (існують безкоштовні версії)	складності одночасної роботи декількох користувачів з однією інформаційною системою
		простота освоєння	підвищений вплив «людського фактора» на коректність таблиць
		гнучкість	складності організації безлічі таблиць в різних варіантах (чорновий, узгоджений, затверджений і т.п.), що вимагає додаткових зусиль по організації роботи з файлами
		потужні інструменти обробки даних (включаючи статистичні функції) і презентацій результатів	складність із забезпеченням конфіденційності даних інформаційної системи. [4].
		практично необмежені можливості з обміну даними з іншими інформаційними системами порівняно низькі витрати на впровадження системи бюджетування і на її модернізацію	низька продуктивність
MathLab	процесор Pentium III, 4, Xeon, Pentium M; AMD Athlon, Athlon XP, Athlon MP	потужність	Невисока інтегрованість середовища
	256 Мбайт оперативної пам'яті (рекомендується 512 Мбайт)	надійність	не надто виразна довідкова система
	400 Мбайт дискового простору (тільки для самої системи MatLab і її Help) [3]	прозорість	
	операційна система Microsoft Windows 2000, (SP3) / XP	універсальність	специфічний редактор коду MatLab-програм
		гнучкість	
		Розширення можливостей досягається за рахунок використання великої кількості спеціально розроблених пакетів розширення, наборів інструментів. [6].	

4

Продовження таблиці 1.

MathCad	процесор Pentium II або вище	природна математична мова	обмежені можливості існуючих операторів
	128 Мбайт оперативної пам'яті (рекомендується 256 Мбайт або більше);	Наочність	труднощі реалізації складних алгоритмів
	200-400 Мбайт дискового простору	Гарна діагностика помилок	
	операційні системи: Windows 98 / Me / NT 4.0 / 2000 / XP	Висока точність обчислень	
		Забезпечення роботи з комплексними числами	
		Реалізація багатьох стандартних функцій обчислювальної математики	
		Можливості символьних математичних перетворень	
Mathematica	процесор Pentium II або вище	вирішення більшості математичних задач в діалоговому режимі без традиційного програмування	незвичайна мова програмування
	128 Мбайт оперативної пам`яті (рекомендується 256 Мбайт або більше)	сильна і вишукана графіка	
	400-550 Мбайт дискового простору		
	операційні системи: Windows 98/Me/ NT 4.0/2000/2003 Server/2003x64/XP/XPx64.		
Maple	процесор Pentium III 650 МГц	здатність виконувати арифметичні дії в символьному вигляді	Дорога вартість
	128 Мбайт оперативної пам'яті (рекомендується 256 Мбайт)	При роботі з дробами і корінням програма виробляє необхідні скорочення і перетворення в стовпчик, що дозволяє уникнути помилок при округленні	Задумливість
	400 Мбайт дискового простору		
	операційні системи: Windows NT 4 (SP5) / 98 / ME / 2000/2003 Server / XP Pro / XP Home		

Діаграма взаємодії користувача з програмою

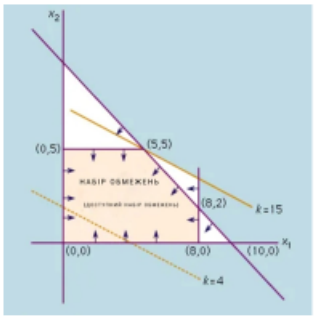


Модель симплекс-метода

4

$F(x_1, \dots x_n) = c_1x_1 + \cdots + c_nx_n$ (1) цільова функція

$$\left\{\begin{array}{l} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n \le b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n \le b_2 \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n \le b_3 \end{array}\right\}$$ (2) Функції обмежень



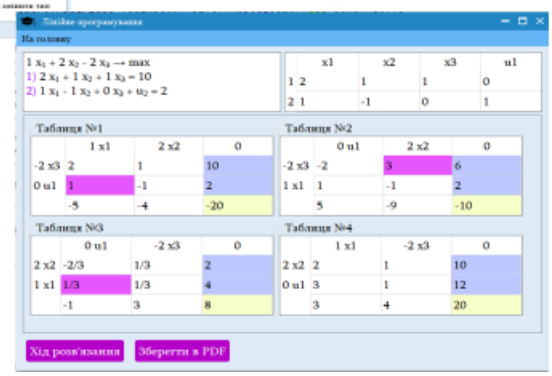
Алгоритм симплекс-методу

7

Введення початкових обмежень

6

Розв'язана задача симплекс-методом



Математична модель транспортної задачі

$$\sum_{j=1}^{m} x_{ij} \leq a_i$$

$$\sum_{i=1}^{n} x_{ij} \leq b_j$$

(3) обмеження транспортної задачі

$$x_{ij} \geq 0, \quad i = 1, n$$

$$u_i + v_j = C_{ij} \text{ для } x_{ij} \geq 0$$

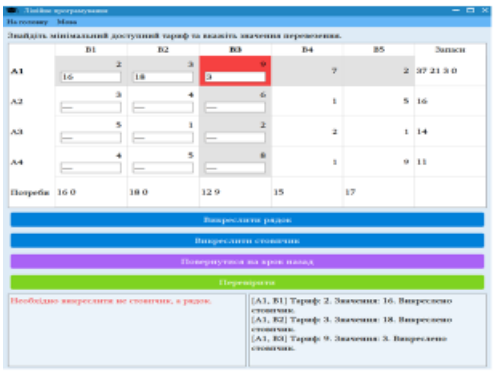
(4) метод потенціалів

$$u_i + v_j = C_{ij} \text{ для } x_{ij} \leq 0$$

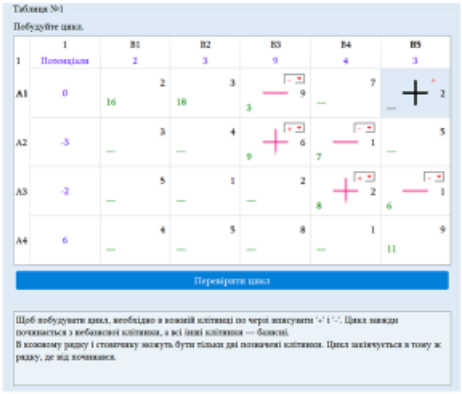
	Місто A	Місто B	Місто C	Пропозиція
Компанія 1	C_{1A}	C_{1B}	C_{1C}	a_1
Компанія 2	C_{2A}	C_{2B}	C_{2C}	a_2
Компанія 3	C_{3A}	C_{3B}	C_{3C}	a_3
Попит	b_1	b_2	b_3	$\Sigma a_i = \Sigma b_i$

Алгоритм транспортної задачі

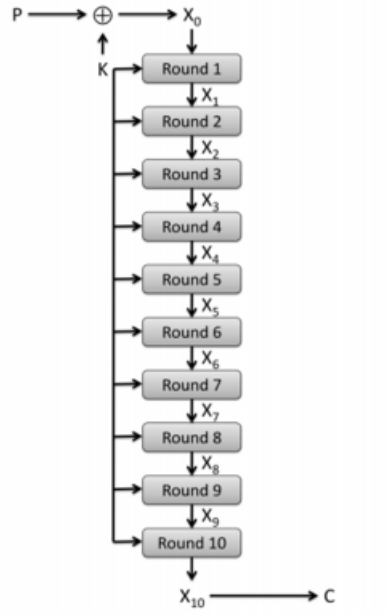
Розподіл ресурсів транспортної задачі



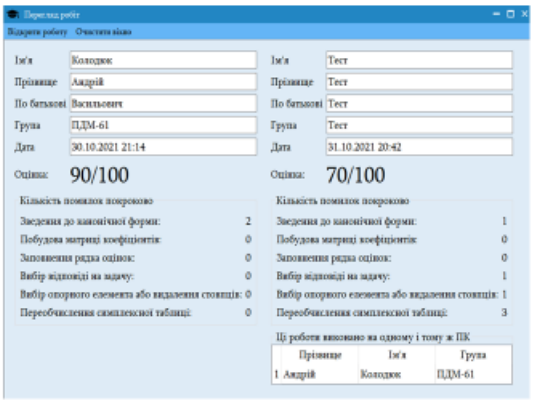
Побудова циклу транспортної задачі



Смехатичне зображення алгоритму AES 12



Форма для перегляду результатів розв'язання



11

13

ОТРИМАНІ РЕЗУЛЬТАТИ РОБОТИ

Науковий результат:
Удосконалення інформаційної технології підвищення ефективності визначення оптимального рішення на основі методів оптимізації.

Практичний результат:
У користувачів є можливість розв'язувати задачі лінійного програмування, такі як "Симплекс-метод" та "Транспортна задача" у програмі із графічним інтерфейсом. Присутній ручний (покроковий) режим із перевіркою усіх дій та автоматичний (миттєва побудова усіх таблиць найкоротшим маршрутом).

Користувач може переглянути результат розв'язання задачі та відправити файл результату викладачеві. Перед зберіганням цей файл шифрується, тому його відредагувати неможливо.