УДК 004.4

Денисенко Б.О., Молчанова М.О., Кліменко В.І.

Хмельницький національний університет

ПІДХІД ДО АВТОМАТИЗОВАНОГО ВИРІШЕННЯ ЗАДАЧ ЛІНІЙНОГО ПРОГРАМУВАННЯ

Розглянуто підхід до автоматизованого вирішення задач лінійного програмування та розглянуто прикладні аспекти відповідного програмного застосунку. Забезпечено можливість розв'язувати різні типи завдань лінійного програмування, обробку великого обсягу вхідних даних, підтримку різноманітних типів задач лінійного програмування, відтворення для знаходження результату.

The approach to automated solution of linear programming problems is considered and the applied aspects of corresponding software application are considered. The ability to solve various types of linear programming tasks, processing of large volume of input data, support for various types of linear programming tasks, reproduction to find the result is provided.

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 [1].

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, при організації функціонування та розвитку соціальних процесів, їх координації з господарськими та економічними процесами [2]. Оптимальні (ефективні) рішення дозволяють досягати мети при мінімальних витратах трудових, матеріальних і сировинних ресурсів.

Ефективність автоматизації за допомогою інформаційних технологій полягає у швидкості та точності самих обчислень. До того ж за допомогою комп'ютера ми матимемо змогу обробляти дійсно великі обсяги даних і проводити складні оптимізаційні розрахунки, що дозволятиме нам значно економити велику кількість часу і та зберігати наші ресурси.

Метою роботи є розробка підходу до автоматизованого вирішення задач лінійного програмування і його апробація. Має бути забезпечена підтримка різноманітних типів задач лінійного програмування, включаючи мінімізацію, максимізацію та цілочисельне програмування.

Для полегшення проєктування підходу було створено декілька діграм. На рисунку 1 подано діаграму варіантів використання [3].

Рисунок 1 – Діаграма варіантів використання застосунку

Зміна кількості «стовбців» чи «рядків» викликає за собою оновлення матриці вхідних даних. «Вхідні дані» містять в собі цільову функцію задачі лінійного програмування, обмеження, а також її тип (максимізація чи мінімізація). Вибір алгоритму включає в себе додаткові покращення для оптимізації, як от цілочисельна оптимізація та інші.

Задля запуску розв'язку обов'язковою умовою є «ініціалізація вхідних даних».

На рисунку 2 за допомогою «Activity diagram»[14] представлено хід розв'язку задачі.

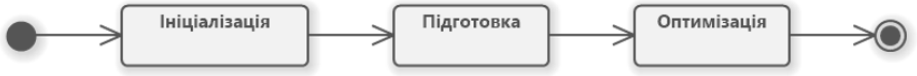

Рисунок 2 – Діаграма активності ходу розв'язування ЗЛП

Процес ініціалізації включає в себе заповнення цільової функції, обмежень, а також типу задачі (мінімум/максимум). Підготовку ж у свою чергу можна розбити на декілька етапів. Перший – приведення задачі до коректного типу (задачу на мінімум перетворити на задачу на максимум) (рисунок 3).

Рисунок 3 – Діаграма активності дочірнього процесу підготовки вхідних даних ЗЛП. Приведення задачі до коректного типу

Наступним кроком буде перетворення обмежень за допомогою внесення додаткових змінних. Схематично зображено на рисунку 4.

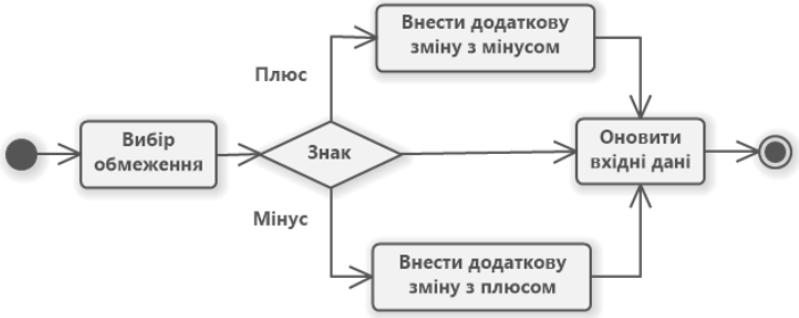

Рисунок 4 – Діаграма активності дочірнього процесу підготовки вхідних даних ЗЛП. Корегування обмежень

Останнім кроком слід перевірити, чи можливо виділити в задачі базис, і якщо ні, додати штучні змінні (рисунок 5).

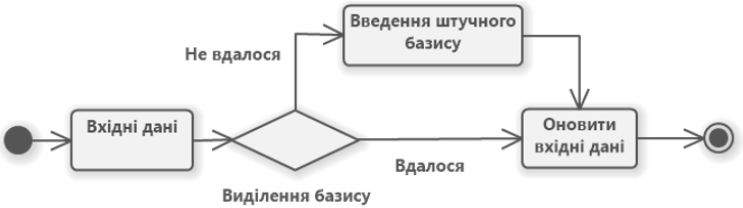

Рисунок 5 – Діаграма активності дочірнього процесу підготовки вхідних даних ЗЛП. Введення штучного базису

Після підготовки даних можна приступати до розв'язку задачі, алгоритм чого зображено на наступній «activity diagram» (рисунок 6).

Рисунок 6 – Діаграма активності процесу оптимізації ЗЛП

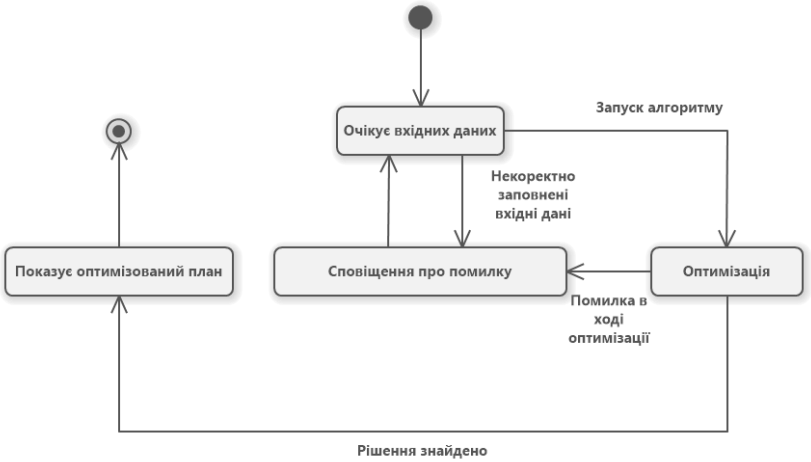

Рисунок 7 – Діаграма станів

Варто звернути увагу на те, що приховано під «додатковими умовами» та «внесенням змін до опрацьованих даних». Якщо взяти, до прикладу, алгоритм розв'язування ЗЛП в цілих числах, то за другим буде сховано додавання нового обмеження, а за першим – перевірка на те, чи можливо взагалі додати те обмеження.

На рисунку 7 зображено діаграму станів розв'язку задачі. Помилками у вхідних даних можуть бути, до прикладу, пусті, або не правильно поля даних (цільова функція, обмеження). У разі їх коректності буде можливо запустити алгоритм оптимізації. У його роботі також можуть статися помилки: не перехоплені на етапі заповнення, або ж викликані в ході роботи (план не можливо оптимізувати). Коли рішення буде все ж таки знайдено, показати його користувачу.

Для того, щоб показати взаємодію деякий об'єктів у динаміці, побудовано діаграму послідовностей (рисунок 8).

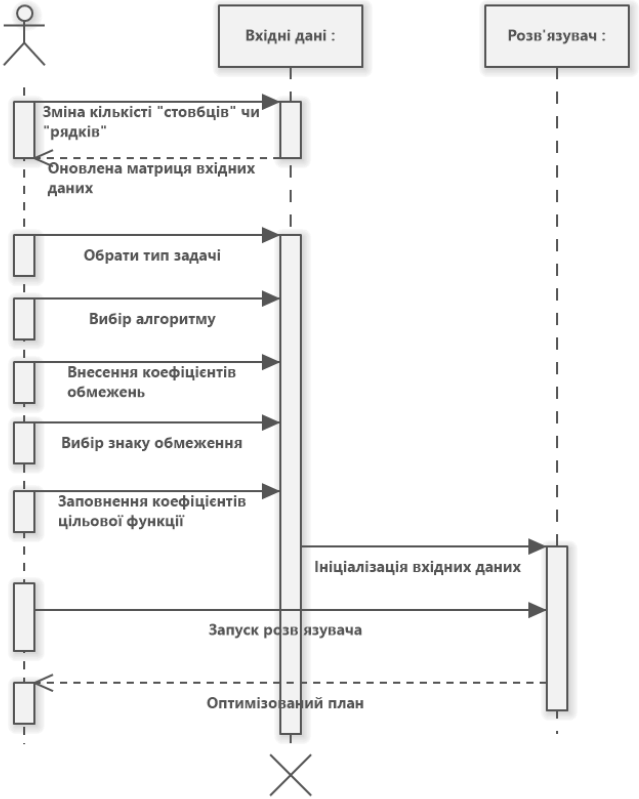

Рисунок 8 – Діаграма послідовностей розв'язування ЗЛП

Серед можливих архітектур для розробки була обрана «компонентна» [4], коли кожен компонент відповідає за конкренту функціональність або аспект застосунку. Така декомпозиція дозволяє логічно групувати функції та розділяти відповідальність. Вибір компонентної архітектури з фокусом на розподілі компонентів по категоріях є хорошою стратегією. Це дозволяє зберегти чіткість, модульність та розширюваність застосунку, а також забезпечити зручний спосіб взаємодії з користувачем.

Для ще більшої упорядкованості, компоненти додатково розподіляються на дві логічні категорії – для власноруч розробленої бібліотеки для вирішення ЗЛП та для взаємодії з користувачем.

Компоненти бібліотеки будуть містити необхідні алгоритми та методи для розв'язання задач, як от симплекс-алгоритм, метод Гоморі [5], введення штучного базису, тощо. Проведені тут розрахунки будуть передаватися іншим компонентам ІС. Компоненти для взаємодії з користувачем будуть відповідати за зовнішній вигляд, включаючи введення даних та відображення результатів. Включає в себе графічний інтерфейс (GUI).

На основі розробленої структури програми було створено діаграму класів, що буде використовуватися при створенні проєкту. Розроблена діаграма зображена на рисунку 9.

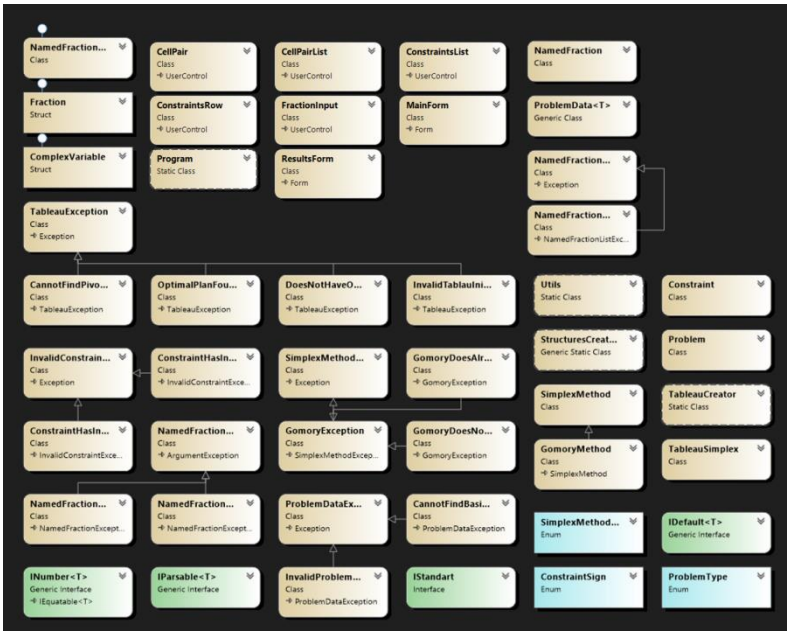

Рисунок 9 – Діаграма класів інформаційної системи

«SimplexMethod» – клас, що відповідає за систему оптимізації задачі лінійного програмування. Також зберігає в собі історію оптимізацій. «GomoryMethod» – його клас-нащадок. Для вирішення задачі в цілих числах.

«TableauSimplex» – являє собою таблицю оптимізацій. «TableauCreator» – конвертує канонічну форму задачі до стандартної. Готує її для «TableauSimplex». «ProblemData» – вхідні дані для таблиці.

«StructuresCreator» – статичний «Generic» клас. Дає змогу створювати масиви різної розмірності з визначеними для них стандартними значеннями від «IDefault», також дає можливість конвертувати масив, елементи якого наслідують інтерфейс «IParsable».

«ComplexVariable» та «Fraction» структури, що відповідають за подання комплексних чисел та дробів відповідно. Обо'є наслідують три інтерфейси: «IDefault», «IParsable» та «INumber». Останній необхідний для порівняння значень в «Generic» методах.

У класі «Utils» містяться методи для комфортного представлення таблиці будь-якої таблиці в вигляді тексту.

«NamedFraction» та «NamedFractionList» відповідають за подання дробів з їх ідентифікаторами як елементів обмежень або цільової функції.

«Problem» та «Constraints» використовуються при прямому звернені до бібліотеки. Запезпечують правильний парсинг стрічових даних для подальшої їх обробки в «TableauCreator» та «TableauSimplex».

Перелічуванні типи даних «ProblemType», «SimplexMethodType» та «ConstraintsSign» використовуються задля визначення типу поставленої задачі, виду екстремумів та знаку обмеження відповідно.

Класи, які на діаграмі розташовані після інтерфейсів та перелічуваних типів даних відповідають за правильне відображення помилок в ході роботи програми. Як от: «TableauException» – загальний клас для помилок в ході обробки (оптимізації) таблиці. «InvalidTablauInitDataException» – некоректно задані вхідні дані таблиці. «CannotFindPivotElementException» – не можливо знайти елемент відносно якого буде здійснюватися оптимізація. «OptimalPlanFoundedException» – не можливо покращити результуюче рішення, оскільки оптимальне вже знайдено. «DoesNotHaveOptimalPlanException» – не можливо знайти оптимальний план. «SimplexMethodException» – загальний клас, що описує помилки, котрі можуть виникнути в ході оптимізації симплексним-методом. «GomoryException» – помилки похідного класу (цілочисельна оптимізація). «GomoryDoesNotHaveIntegerSolutionException» – не можливо знайти цілочисельне рішення задачі. «GomoryDoesAlreadyHaveIntegerSolutionException» – пошук цілочисельного рішення не можливий, оскільки воно вже знайдено. «ProblemDataException» – помилка вхідних даних таблиці. «InvalidProblemDataInitialParametersException» –некоректно задані вхідні параметри. «CannotFindBasisException» – не може віднайти базис. Показує також кількість знайдених, та кількість необхідних векторів до утворення базису. Використовується

задля майбутнього конвертування вхідних даних задачі з внесенням штучного базису. «InvalidConstraintException» – некоректні вхідні дані обмеження. «ConstraintHasInvalidBValueException» – помилки в задані обмеження. «ConstraintHasInvalidSignException» – помилковий знак в обмеженні. «NamedFractionException» – загальний клас для помилок виявлених в ході конвертування числа з ідентифікатором. «NamedFractionNumberException» – некоректно задано число. «NamedFractionNameException» – некорекний ідентифікатор. «NamedFractionNameIsAlreadyExistsException» – у одному й тому ж самому обмежені ідентифікатор не може зустрічатися більше двох разів. «NamedFractionListException» – помилка конвертування стрічки чисел з їх ідентифікатором.

«MainForm» – основне вікно програми. Задаються всі вхідні дані. «ResultForm» – необхідний для виводу результату оптимізація з усіма пройденими кроками. «ConstraintsList», «CellPairList», «FractionInput», «ConstraintsRow» та «CellPair» користувацькі компоненти, значно полегшують розробку.

В усіх вище згаданих реченнях стосовно помилок, ідентифікатор слід сприймати як унікальну змінну, що стоїть біля числа.

Після введення початкових даних та натискання кнопки «Solve» програма вирішує задану йому задачу лінійного програмування (рисунок 10). Далі розв'язок виводиться на екран користувача (рисунок 11).

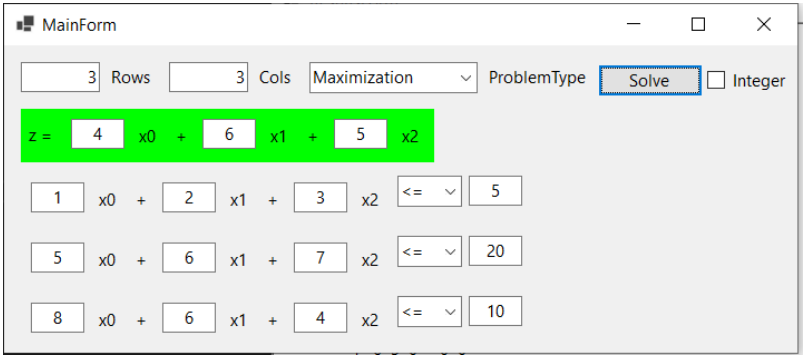

Рисунок 10 – Головний екран застосунку

У результаті виконання завдання було розроблено підхід і відповідний застосунок для вирішення задач лінійного програмування, який виконує всі необхідні функції. Система може ефективно розв'язувати різні типи завдань лінійного програмування, працює офлайн, обробляє велику кількість вхідних даних, підтримує різноманітні типи задач лінійного програмування та показує кроки до знаходження результату.

Щодо перспектив та ступеня впровадження розробленої системи – його можливості і переваги, дають йому потенціал для широкого використання в

академічних дослідженнях, промисловості та інших галузях, де потрібно розв'язувати задачі лінійного програмування.

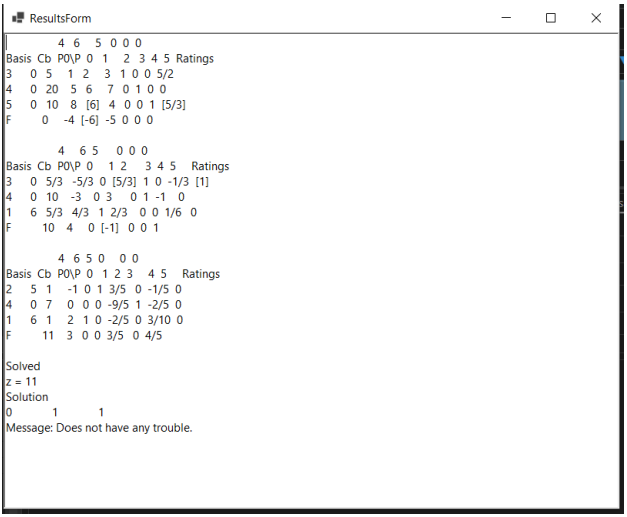

Рисунок 11 – Результат виконання програми

У майбутньому планується розширення функціональності: додавання нових типів задач лінійного програмування, підтримки різних алгоритмів розв'язування, вдосконалення інтерфейсу користувача тощо; аналіз та вдосконалення алгоритмів і структур даних для покращення швидкості розв'язування задач та зменшення використання ресурсів; додавання налаштувань та параметрів, які дозволять користувачам налаштовувати програмний застосунок для виконання конкретних завдань. Ці можливості та шляхи до вдосконалення сприятимуть подальшому успіху та використанню розробленої системи.

Перелік посилань

1. О. О. Ємець, О. С. Пічугіна, О. Б. Маций, К. П. Коробчинський. Лінійне програмування. Навчальний посібник. Х. : ХНАДУ, 2019. 102 с.
2. А. А. Яровий, Л. М. Ваховська, Л. В. Крилик. Математичні методи дослідження операцій. Лінійне програмування. Частина 1. Навчальний посібник. Вінниця: ВНТУ. 2020. 86 с.
3. eMathHelp. Simplex Method Calculator. URL: https://www.emathhelp.net/en/calculators/linear-programming/simplex-method-calculator.
4. What is Component-Based Architecture? URL: https://www.mendix.com/blog/what-is-component-based-architecture/
5. Метод Гоморі (метод відсікаючих площин). URL: https://www.mathros.net.ua/metod-gomori-metod-vidsikajuchyh-ploshhyn.html