МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Чорноморський національний університет
імені Петра Могили
Факультет комп'ютерних наук
Кафедра інтелектуальних інформаційних систем

ДОПУЩЕНО ДО ЗАХИСТУ
Завідувач кафедри інтелектуальних
інформаційних систем, д-р техн. наук, проф.
_____Ю. П. Кондратенко
«____»_____2024 р.

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

ПЛАНУВАННЯ ТА ОПТИМІЗАЦІЯ ТРАНСПОРТНИХ ВАНТАЖОПЕРЕВЕЗЕНЬ З НЕЧІТКИМИ ВИТРАТАМИ

Спеціальність 122 «Комп'ютерні науки»

122 – КРБ – 401.22010702

Виконав студент 4-го курсу, групи 401
_____ С. С. Безощук
«17» червня 2024 р.

Керівник: ст. виклад. кафедри ІПЗ
_____С. Ю. Боровльова
«17» червня 2024 р.

Миколаїв – 2024

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Чорноморський національний університет ім. Петра Могили
Факультет комп'ютерних наук
Кафедра інтелектуальних інформаційних систем

Рівень вищої освіти **бакалавр**
Спеціальність **122 «Комп'ютерні науки»**
(шифр і назва)
Галузь знань **12 «Інформаційні технології»**
(шифр і назва)

ЗАТВЕРДЖУЮ

Завідувач кафедри інтелектуальних
інформаційних систем, д-р техн. наук, проф.
_____ Ю. П. Кондратенко
«____» _____20___р.

З А В Д А Н Н Я
на виконання кваліфікаційної роботи

Видано студенту групи 401 факультету комп'ютерних наук Безощуку Станіславу Сергійовичу.

1. Тема кваліфікаційної роботи «Планування та оптимізація транспортних вантажоперевезень з нечіткими витратами».

Керівник роботи Боровльова Світлана Юріївна, ст. виклад. кафедри ІПЗ.

Затв. наказом Ректора ЧНУ ім. Петра Могили від «28» грудня 2023 р. № 271

2. Строк представлення кваліфікаційної роботи студентом «18» червня 2024 р.

3. Вхідні (початкові) дані до роботи: функціональні вимоги до програмного забезпечення з планування та оптимізації транспортних вантажоперевезень з нечіткими витратами.

Очікуваний результат: програмний застосунок для планування та оптимізації маршрутів транспортних вантажоперевезень.

4. Перелік питань, що підлягають розробці (зміст пояснювальної записки):

- аналіз предметної області та аналогічного програмного забезпечення;
- встановлення вимог та формування технічного завдання на основі здійсненного аналізу;

– аналіз методів та алгоритмів, необхідних, для розробки програмного забезпечення;

– реалізація та тестування застосунку.

5. Перелік графічного матеріалу: 22 рисунки, 12 таблиць і презентація.

6. Завдання до спеціальної частини: «Питання охорони праці стосовно транспортних вантажоперевезень »

7. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис
Спеціальна частина з охорони праці	Алексєєва А. О., доцент кафедри екології	

Керівник роботи __ст. викладач кафедри ІПЗ Боровльова С. Ю._____
(наук. ступінь, вчене звання, прізвище та ініціали)

_____
(підпис)

Завдання прийнято до виконання ___Безощук С. С._____
(прізвище та ініціали)

_____
(підпис)

Дата видачі завдання «_14_» _____січня____ 2024 р.

КАЛЕНДАРНИЙ ПЛАН
виконання кваліфікаційної роботи

Тема: «Планування та оптимізація транспортних вантажоперевезень з нечіткими витратами»

№	Найменування роботи	Початок	Закінчення	Примітки
1.	Подання заяви на затвердження теми та керівників КРБ	10.11.2023	15.11.2023	Виконано
2.	Отримання завдання на виконання КРБ	10.01.2024	15.01.2024	Виконано
3.	Складання календарного плану роботи на весь період виконання КРБ	16.01.2024	30.01.2024	Виконано
4.	Отримання завдання на переддипломну практику	15.04.2024	29.04.2024	Виконано
5.	Проходження переддипломної практики, збір та аналіз матеріалів до КРБ	29.04.2024	11.05.2024	Виконано
6.	Розробка звіту з переддипломної практики	12.05.2024	15.05.2024	Виконано
7.	Виконання КРБ: аналіз сучасного стану планування та оптимізація транспортних вантажоперевезень з нечіткими витратами огляд існуючих технологій, розробка ПЗ	13.05.2024	22.06.2024	Виконано
8.	Перший попередній захист КРБ на засіданні комісії кафедри	27.05.2024	27.05.2024	Виконано
9.	Доробка та остаточне оформлення КРБ	28.05.2024	09.06.2024	Виконано

10.	Другий попередній захист КРБ на засіданні комісії кафедри	10.06.2024	10.06.2024	Виконано
11.	Подання КРБ рецензенту	13.06.2024	13.06.2024	Виконано
12.	Подання КРБ, її електронної копії та інших документів (відгуку, рецензії) до захисту	17.06.2024	21.06.2024	Виконано
13.	Захист БКР перед екзаменаційною комісією (ЕК)	24.06.2024	28.06.2024	

Розробив студент _____Безощук С. С._____ _____
(прізвище та ініціали) (*підпис*)

Керівник роботи ст. виклад. кафедри ІПЗ Боровльова С. Ю. _____
(наук. ступінь, вчене звання, прізвище та ініціали) *(підпис)*

«__29__» _____січня_____ 2024 р.

АНОТАЦІЯ

кваліфікаційної роботи студента групи 401 ЧНУ ім. Петра Могили

Безощука Станіслава Сергійовича

Тема: «Планування та оптимізація транспортних вантажоперевезень з нечіткими витратами»

Програмне забезпечення для планування та оптимізації транспортних вантажоперевезень широко використовуються в логістичних компаніях. Завдяки даним застосункам, надається можливість обрати оптимальний маршрут для збереження ресурсів та часу.

Об’єктом роботи являються процеси транспортних вантажоперевезень в умовах невизначеності. Предметом роботи є методи та підходи для планування і оптимізації транспортних вантажоперевезень. Метою роботи являється планування і оптимізація транспортних вантажоперевезень з нечіткими витратами шляхом створення відповідного програмного застосунку.

У першому розділі описані проблеми транспортних вантажоперевезень як у світі, так і в Україні; проаналізовано існуючі аналоги та на їх прикладі поставлено задачу з відповідними вимогами. У другому розділі представлено методи й алгоритми планування та оптимізації транспортних вантажоперевезень, використані при розробці програмного забезпечення. У третьому розділі описано процес розробки застосунку та його тестування. Спеціальна частина з охорони праці, що повністю, пов’язана з проблемами транспортних вантажоперевезень.

В результаті виконання кваліфікаційної роботи бакалавра реалізовано програмне забезпечення для планування та оптимізації транспортних вантажоперевезень з нечіткими витратами.

КРБ містить 56 сторінок, 5 додатків, 22 рисунки, 12 таблиць і посилання на 23 джерела.

Ключові слова: оптимізація транспортних вантажоперевезень, нечіткі витрати, метод потенціалів, метод Фогеля, Windows Forms, C#.

ABSTRACT

to the qualification work by the student of the group 401 of Petro Mohyla Black Sea National University

Bezoshchuk Stanislav

“Planning and optimization of freight transport with fuzzy costs”

Software for planning and optimizing transport and cargo transportation is widely used in logistics companies. Thanks to these applications, it is possible to choose the optimal route to save resources and time.

The object of the work is the processes of transport and cargo transportation in conditions of uncertainty. The subject of the work is methods and approaches for planning and optimization of transport and cargo transportation. The purpose of the work is the planning and optimization of freight transportation with uncertain costs by creating a suitable software application.

The first chapter describes the problems of freight transportation both in the world and in Ukraine; existing analogues were analyzed and a task with appropriate requirements was set on their example. The second chapter presents the methods and algorithms for planning and optimization of transport and cargo transportation, used in the development of software. The third section describes the application development process and its testing. The special part is entirely related to the problems of transport and freight transportation.

As a result of completing the bachelor's qualification work, software for planning and optimizing transport and freight transportation with uncertain costs was implemented.

The work contains 56 pages, 5 appendices, 22 figures, 12 tables and 23 sources in it.

Key words: optimization of freight transport, fuzzy costs, method of potentials, Vogel's method, Windows Forms.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ 10

ВСТУП 11

1 АНАЛІЗ СТАНУ ТА ПРОБЛЕМ ТРАНСПОРТНИХ ВАНТАЖОПЕРЕВЕЗЕНЬ 13

1.1 Фактори впливу на транспортні вантажоперевезення 13

1.2 Проблема оптимізації транспортних вантажоперевезень 19

1.3 Аналіз аналогів планування та оптимізації транспортних вантажоперевезень з використанням нечітких витрат 22

1.4 Постановка задачі 24

Висновки до розділу 1 26

2 ПІДБІР МЕТОДІВ ДЛЯ ВИРІШЕННЯ ПОСТАВЛЕНОЇ ЗАДАЧІ 27

2.1 FuzzyTOPSIS 27

2.2 Метод північно-західного кута 29

2.3 Метод Фогеля 31

2.4 Метод потенціалів 34

Висновки до розділу 2 39

3 РЕАЛІЗАЦІЯ ПРОГРАМНОГО ЗАСТОСУНКУ 41

3.1 Обґрунтування вибору технологій розробки 41

3.2 Опис компонентів застосунку 42

3.3 Проектування користувацького інтерфейсу 48

3.4 Програмування основної логіки застосунку 50

3.5 Тестування застосунку 56

Висновки до розділу 3 61

ВИСНОВКИ 62
ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ 63
ДОДАТОК А Реалізація сутності нечітких витрат на мові C# 66
ДОДАТОК Б Реалізація методу Фогеля на мові C# 67
ДОДАТОК В Реалізація методу потенціалів на мові C# 70
ДОДАТОК Г Вміст Form1.cs 75
ДОДАТОК Д Вміст MatrixForm.cs 79

ПЕРЕЛІК СКОРОЧЕНЬ

AI – Artificial Intelligence

FTPS – Fuzzy Transportation Problem Solvers

GPS – Global Positioning System

GUI – Graphical User Interface

IDE – Integrated Development Environment

IoT – Internet of Things

IT – Information Technology

TOPSIS – Technique for Order Preference by Similarity to Ideal Solution

ВСТУП

З розвитком глобалізації та зростанням обсягів виробництва і торгівлі, ефективне планування та оптимізація транспортних вантажоперевезень стають критично важливими завданнями для підприємств у всіх галузях економіки. Заходи з підвищення ефективності транспортних процесів є стратегічними для забезпечення конкурентоспроможності компаній та виконання їхніх бізнес-цілей.

Дана кваліфікаційна робота бакалавра спрямована на вивчення та аналіз сучасних методів та інструментів планування й оптимізації транспортних вантажоперевезень з метою підвищення їхньої ефективності та економічної доцільності.

Пандемія COVID-19 показала наскільки важливо правильно планувати маршрут як для транспортування необхідних ресурсів, так задля перевезення хворих людей. Також це впливало на сам ринок, через що, страждали підприємства.

Тема, наразі, є актуальною, особливо, через російське вторгнення в Україну, що ставить під загрозу український суверенітет. І тому, дуже важливо, вчасно доставляти озброєння та гуманітарну допомогу у відповідні регіони держави задля урегулювання становища як на лінії фронту, так і в населених пунктах.

Об’єктом роботи являються процеси транспортних вантажоперевезень в умовах невизначеності.

Предметом роботи є методи та підходи для планування і оптимізації транспортних вантажоперевезень.

Метою роботи являється планування і оптимізація транспортних вантажоперевезень з нечіткими витратами шляхом створення відповідного програмного застосунку.

Для досягнення поставленої мети необхідно виконати наступні завдання:

- аналіз сучасного стану та проблем транспортних вантажоперевезень;
- методи планування та оптимізації транспортних процесів;

– розробка програмного застосунку оптимізації вантажоперевезень з урахуванням різноманітних обмежень та вимог;

– тестування розроблених інструментів на прикладі реальних виробничих ситуацій.

Результати даної роботи можуть стати важливим внеском у практику управління транспортними вантажоперевезеннями, сприяючи підвищенню ефективності, зниженню витрат та підвищенню конкурентоспроможності підприємств.

1 АНАЛІЗ СТАНУ ТА ПРОБЛЕМ ТРАНСПОРТНИХ ВАНТАЖОПЕРЕВЕЗЕНЬ

1.1 Фактори впливу на транспортні вантажоперевезення

Транспортні вантажоперевезення - це надзвичайно складний і деталізований процес, що охоплює величезну мережу взаємодіючих факторів, які впливають на його ефективність та результативність. Цей процес включає в себе взаємодію між різними галузями, технологіями, людьми та середовищем, і кожен з цих аспектів має вирішальне значення для успішного виконання вантажних перевезень.

Поглиблене розуміння транспортних вантажоперевезень вимагає аналізу різноманітних факторів, починаючи від економічних умов та закінчуючи соціокультурними та екологічними аспектами.

1.1.1 Економічні фактори

Економічний контекст визначає не лише обсяги вантажоперевезень, а й шляхи їхнього розвитку. При зростанні економіки спостерігається збільшення споживчої активності, що породжує підвищений попит на перевезення товарів. Особливо це актуально для розвинених країн, де високий рівень доходу веде до підвищення споживчого попиту. Такий попит у свою чергу стимулює ріст обсягів вантажних перевезень, що стає суттєвим критерієм для оцінки стану економіки.

Рисунок 1.1 – Вартість бензину в Україні за роками [1]

Однак, коли мова йде про витрати на транспортування, одним із ключових факторів є ціна пального. Підвищення цін на пальне призводить до зростання витрат перевізників (рис. 1.1), оскільки вони змушені збільшувати витрати на заправку своїх транспортних засобів. Це, в свою чергу, може призвести до зростання цін на транспортні послуги.

Крім того, важливим аспектом також є вартість обслуговування транспортних засобів. Сюди входять витрати на ремонт, технічне обслуговування та страхування. Ці витрати становлять суттєву частину загальних витрат перевізників і можуть впливати на остаточну вартість транспортних послуг для споживачів. Таким чином, економічний контекст має безпосередній вплив на витрати та ціни у сфері транспортних перевезень.

1.1.2 Технологічні фактори

Завдяки аналітиці даних можна вдосконалити маршрутизацію та вибір оптимальних транспортних засобів для конкретних вантажів. Аналізуючи великі обсяги даних про трафік, погодні умови, витрати на пальне та інші фактори, логістичні підрозділи можуть приймати більш обґрунтовані рішення щодо вибору маршрутів та транспортних засобів. Технології аналізу даних також дозволяють прогнозувати

можливі проблеми та ризики на шляху, допомагаючи уникнути затримок та оптимізувати час доставки.

Використання передових технологій, таких як системи GPS, IoT, хмарні технології та інші інновації, стало необхідним елементом в оптимізації логістичних процесів. Системи GPS дозволяють точно визначати місцезнаходження транспортних засобів у реальному часі, що полегшує відстеження вантажів та планування оптимальних маршрутів [2]. IoT датчики можуть надавати важливі дані про стан вантажу, температуру, вологість та інші параметри, що допомагає у забезпеченні якості та безпеки перевезень [3]. Хмарні технології забезпечують доступ до даних з будь-якого місця, сприяючи спільній роботі та координації всіх учасників логістичного ланцюжка [4].

1.1.3 Політичні та правові фактори

Законодавство щодо безпеки та технічного стану транспортних засобів також може впливати на логістику та вибір маршрутів. Правила дорожнього руху, вимоги до дозволів та стандарти безпеки можуть варіюватися в залежності від країни чи регіону, що може впливати на швидкість перевезень та вибір оптимальних маршрутів.

Політика уряду стосовно митних тарифів та імпортних обмежень може мати значний вплив на вартість перевезень. Зміни в тарифах або митах можуть призвести до збільшення витрат на міжнародні перевезення, оскільки ці платежі додаються до загальних витрат компаній на логістику.

Угоди між державами щодо міжнародних перевезень можуть встановлювати тарифи, правила та умови для транспортування товарів через кордони. Такі угоди можуть сприяти зниженню тарифів або спрощенню митних процедур, що полегшує та здешевлює міжнародну торгівлю. Однак, недотримання угод або введення нових тарифів може призвести до ускладнень та збільшення витрат на логістику.

1.1.4 Соціокультурні фактори

Зміни у розмірі населення впливають на динаміку вантажних потоків та їх характер. При збільшенні населення в певному регіоні, зазвичай, зростає попит на товари та послуги, що призводить до збільшення обсягів перевезень. Це може вимагати більш ефективних логістичних рішень, таких як оптимізація маршрутів транспортних вантажоперевезень, чи розширення складських потужностей. З іншого боку, зменшення населення може призвести до зменшення попиту та обсягів перевезень.

Крім того, зміни в споживчих звичках можуть суттєво впливати на вантажні потоки. Наприклад, популярність певних товарів або послуг може зростати, що призводить до збільшення обсягів їх перевезень. Такі зміни можуть вимагати реорганізації логістичних процесів та впровадження нових стратегій доставки, щоб забезпечити ефективне задоволення попиту.

Таким чином, як населення, так і споживчі звички впливають на обсяги та напрямки вантажних перевезень і вимагають від компаній у сфері транспорту і логістики постійного аналізу та адаптації до змін на ринку.

1.1.5 Екологічні фактори

Законодавство, спрямоване на регулювання викидів шкідливих речовин та встановлення обмежень на використання палива, є ключовим фактором, що визначає екологічну політику та впливає на вибір транспортних засобів та маршрутів. У багатьох країнах існують строгі нормативи, спрямовані на зменшення викидів транспортних засобів, особливо у міських районах, де концентрація забруднюючих речовин найвища.

Наприклад, деякі держави встановлюють обмеження на використання дизельних автомобілів через їх великий вплив на якість повітря [5]. Ці обмеження можуть змусити логістичні компанії переглянути свою флоту та шукати більш екологічні альтернативи, такі як електричні або гібридні автомобілі. Крім того, вони можуть стимулювати розвиток нових технологій та інфраструктури для підтримки

більш стійких форм транспорту, наприклад, зарядних станцій для електромобілів або маршрутів для велосипедистів та пішоходів.

Такі екологічні обмеження також можуть призвести до перегляду маршрутів та оптимізації логістичних процесів з метою зменшення викидів та покращення ефективності перевезень. Наприклад, компанії можуть розглядати альтернативні маршрути з меншим трафіком або з використанням більш екологічних видів транспорту, щоб знизити вплив на навколишнє середовище та відповідати вимогам законодавства.

1.1.6 Інфраструктурні фактори

Також важливим аспектом є якість інфраструктури, такої як дороги, мости, залізничні колії, аеропорти та порти, яка в значній мірі визначає ефективність та вартість транспортних вантажоперевезень. Якщо інфраструктура перебуває у поганому стані, це може призвести до численних проблем, таких як затримки у доставці, збільшення ризику аварій та погіршення якості послуг. Наприклад, поганий стан доріг спричиняє збільшення споживання пального та швидше зношування автомобільних шин, що призводить до збільшення загальних витрат на перевезення та тривалості транспортування. Тому важливою метою для України є не лише модернізація, але й постійний моніторинг та обслуговування існуючої інфраструктури з метою підвищення її якості та надійності.

1.1.7 Конкурентні фактори

Дії конкурентів та їх стратегії впливають на логістичні рішення, оскільки вони конкурують за сприйняття клієнтів і можуть пропонувати подібні послуги за різними цінами або з різним рівнем якості. Наприклад, якщо конкуренти пропонують аналогічні послуги за більш низькими цінами, це може змусити логістичні компанії адаптуватися, знижуючи власні ціни, щоб залишитися конкурентоспроможними.

Прийняття рішень у сфері логістики зазвичай враховує конкурентне середовище, в якому діють компанії. Це може означати не лише конкуренцію за цінами, а й змагання за якість обслуговування та інноваційність. Наприклад, якщо один конкурент впроваджує нові технології для покращення ефективності перевезень або підвищення рівня обслуговування, інші компанії можуть бути змушені реагувати, щоб залишитися в грі.

Таким чином, дії конкурентів, їх цінові стратегії та якість обслуговування є ключовими факторами, що впливають на прийняття логістичних рішень. Компанії в сфері логістики повинні постійно аналізувати ринкові тенденції та реагувати на зміни в конкурентному середовищі, щоб успішно конкурувати і задовольняти потреби клієнтів.

1.2 Проблема оптимізації транспортних вантажоперевезень

1.2.1 Ситуація у Світі

Проблема оптимізації транспортних вантажоперевезень є важливою у багатьох країнах світу через зростання обсягів вантажних перевезень, зміни в торговельних потоках, розвиток технологій та інші фактори. Оптимізація вантажоперевезень має на меті забезпечити ефективне використання ресурсів (таких як транспортні засоби, інфраструктура, людські ресурси тощо) з мінімізацією витрат та максимізацією продуктивності.

Основні проблеми оптимізації транспортних вантажоперевезень включають:

– недостатність інфраструктури: деякі регіони можуть стикатися з нестачею доріг, портів або залізничних шляхів, що ускладнює перевезення вантажів;

– затори: надмірний трафік та затори можуть призводити до затримок у доставці вантажів, що збільшує витрати та погіршує обслуговування клієнтів (рис. 1.2) [6];

– неефективне використання транспортних засобів: порожній хід транспортних засобів або недостатня завантаженість вантажів можуть призводити до зайвих витрат та негативно впливати на середовище;

– високі витрати на паливо та енергію: ефективне використання ресурсів, таких як паливо, є важливим для зменшення витрат і впливу на навколишнє середовище [5];

– неоптимальні маршрути: вибір найбільш оптимальних маршрутів для доставки вантажів може бути складним через різні фактори, такі як дорожні умови, погода, вартість проїзду тощо;

– нестабільність в торговельних відносинах: політичні або економічні фактори можуть призводити до змін у торговельних потоках, що ускладнює планування та оптимізацію вантажоперевезень.

Рисунок 1.2 – Затор у Києві, в якому застрягло приблизно 1000 фур [6]

Для вирішення цих проблем використовуються різні підходи, включаючи використання IoT, AI, програмні застосунки для оптимізації вантажоперевезень, а також вдосконалення самих логістичних процесів та інфраструктури [3]. Важливою також є співпраця між урядовими органами, приватним сектором та іншими зацікавленими сторонами для досягнення ефективних рішень у сфері транспортних вантажоперевезень.

1.2.2 Ситуація в Україні

Україна, як і багато інших країн, стикається зі складнощами в оптимізації транспортних вантажоперевезень. Українсько-російська війна призвели до обмежень у торговельних відносинах та ускладнили транспортні маршрути. Крім того, інфраструктура, зокрема стан доріг та залізничної мережі, потребує серйозної модернізації для забезпечення ефективного перевезення вантажів (рис. 1.3). Неоптимальне використання ресурсів, високі витрати на пальне та енергію, а також бюрократичні перешкоди додатково ускладнюють ситуацію [7]. Однак Україна має потенціал для вдосконалення своїх транспортних вантажоперевезень шляхом впровадження новітніх технологій [8]. Адже застосування сучасних технологій у логістиці, таких як системи відстеження та управління вантажами, допомагає підвищити ефективність та контроль над перевезеннями [4]. Україна також має можливість залучення інвестицій для покращення інфраструктури та впровадження

інноваційних рішень у сфері транспортування вантажів, що сприятиме розвитку економіки та підвищенню конкурентоспроможності країни на міжнародному ринку. реформування інфраструктури та співпраці з міжнародними партнерами [9].

Рисунок 1.3 – Стан дорожнього покриття в Україні [10]

Легенда карти:

– відмінна дорога з відносно рівним дорожнім полотном (можливі невеликі ями та нерівності);

– дорога у задовільному стані (можливі глибокі ями, погані відрізки, колія);

– дорога із значними руйнуваннями дорожнього полотна, постійні ями;

– дорога без відгуків.

Важливим кроком для оптимізації транспортних вантажоперевезень в Україні є розвиток мультимодальних перевезень. Залізниця та морські порти України грають ключову роль у міжнародних транспортних відносинах, сприяючи ефективному

перевезенню вантажів [8]. Наприклад, розвиток контейнерних терміналів в портах дозволяє забезпечити швидке та зручне перевезення товарів [11].

1.3 Аналіз аналогів планування та оптимізації транспортних вантажоперевезень з використанням нечітких витрат

1.3.1 Fuzzy Logic Toolbox

Fuzzy Logic Toolbox є програмним забезпеченням, яке часто використовується в середовищі MATLAB для моделювання та симуляції систем на основі нечіткої логіки. Він надає набір інструментів та функцій для створення нечітких систем виведення, контролерів на основі нечіткої логіки та інших застосувань. За допомогою цього інструментарію розробники можуть моделювати та симулювати системи, які включають невизначеність, неоднозначність та нечітку інформацію, що робить його підходящим для різних сфер, включаючи системи керування, прийняття рішень та впізнавання образів [12].

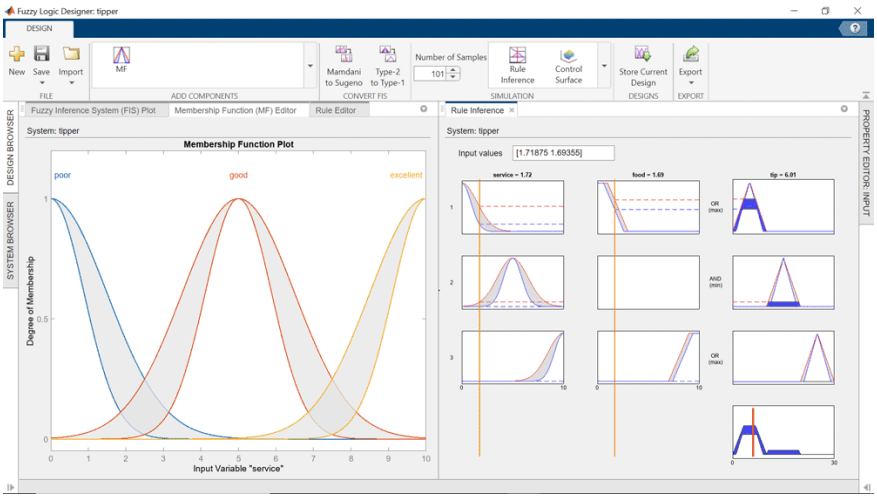

Рисунок 1.4 - Fuzzy Logic Toolbox в середовищі MATLAB [12]

Fuzzy Logic Toolbox у MATLAB надає інструменти для розробки, аналізу та моделювання систем нечіткої логіки. За допомогою цього інструментарію можна моделювати та симулювати системи, які включають невизначеність, неоднозначність

та нечітку інформацію, що робить його підходящим для різних сфер, включаючи системи керування, прийняття рішень та впізнавання образів [12]. Основні можливості Fuzzy Logic Toolbox включають [13]:

– розробка систем нечіткої логіки: інтерфейс для створення і редагування нечітких систем. Підтримка Mamdani та Sugeno типів нечітких систем. Інтерактивне налаштування вхідних та вихідних змінних, визначення термів з використанням функцій належності;

– побудова функцій належності: трикутні, трапецієподібні, гауссові та інші типи функцій. Інтерактивні інструменти для налаштування параметрів функцій належності;

– правила нечіткої логіки: інтерфейс для створення та редагування правил нечіткої логіки. Графічне представлення правил і їхнього впливу на вихідні значення;

– аналіз і тестування: інструменти для візуалізації роботи системи, включаючи поверхневі графіки, що показують залежність вихідних значень від вхідних. Можливість тестування системи на наборі даних для оцінки її точності та продуктивності;

– інтеграція з MATLAB: можливість використання мовою програмування MATLAB для автоматизації створення та налаштування нечітких систем. Інтеграція з Simulink, що дозволяє використовувати нечіткі системи у моделях динамічних систем;

– оптимізація: використання оптимізаційних алгоритмів для налаштування параметрів функцій належності та правил для поліпшення продуктивності системи.

1.3.2 Fuzzy Transportation Problem Solvers

Fuzzy Transportation Problem Solvers - це програмні засоби, призначені для вирішення транспортних проблем у логістиці та управлінні ланцюгами постачання за допомогою технік нечіткої логіки. Вони дозволяють оптимізувати розподіл товарів від джерел до пунктів призначення з урахуванням невизначеності та нечітких даних. Однак, конкретного розробника чи компанію, яка б була відомою за створення саме

цього інструменту, визначити складно. FTPS є узагальненою назвою, яку можуть використовувати різні дослідники та розробники для своїх версій програмних засобів, що вирішують транспортні задачі з використанням нечіткої логіки.

FTPS використовують нечіткі числа для моделювання невизначеності в обсягах, попиті та витратах транспортування. Це дозволяє більш точно відобразити реальні умови. Також дані інструменти можуть використовувати різні методи для вирішення задач, такі як метод центрів тяжіння, інтервальних нечітких чисел або метод альфа-рівнів. Вибір методу залежить від конкретної задачі та доступних даних. Часто в FTPS впроваджують аналіз чутливості для перевірки стійкості отриманих рішень до змін вхідних даних.

1.4 Постановка задачі

1.4.1 Вимоги до проєкту

Основною вимогою є реалізація програмного застосунку для планування та оптимізації транспортних вантажоперевезень з нечіткими витратами. Сам застосунок повинен мати відповідний реалізований функціонал, інтуїтивно зрозумілий інтерфейс та правильно робити розрахунки з даними, які надає користувач, використовуючи відповідні методи та алгоритми.

Маючи стислі вимоги до програмного застосунку, описані вище, можна більш детально описати його вид та функціонал в технічному завданні. В технічному завданні слід врахувати вимоги до продуктивності, масштабованість, стійкість до збоїв та інші важливі аспекти розробки та експлуатації застосунку.

1.4.2 Технічне завдання

Даний, як і будь який інший застосунок, повинен мати назву, за якою його можна буде легко ідентифікувати серед інших продуктів.

Назва продукту: **FuzzyFreightOptimizer**.

Мета розробки: з існуванням бібліотек на різних мовах програмування, з відповідним функціоналом не кожен може ними скористатися, адже не являється спеціалістом в ІТ сфері і тому, це є основною метою розробки даного застосунку – надати змогу застосовувати такі бібліотеки у вигляді ПЗ з, дружнім до новачків, інтерфейсом.

Технічні вимоги: основними вимогами то функціональної складової застосунку можна віднести:

– можливість створення плану перевезень, на основі введених користувачем даних;

– можливість введення нечітких значень транспортних витрат;

– виведення результатів користувачу;

– збереження результатів роботи в якості .txt файлу;

– розробити застосунок таким чином, щоб була можливість додавати нові методи оптимізації (масштабованість).

Маючи сформовані технічні умови, можна перейти до вимог користувацького інтерфейсу. Він повинен виглядати простим, щоб кожен міг на інтуїтивному рівні працювати з застосунком. Кожна частина функціоналу повинна бути логічно згрупована (критерії в одному контейнері, методи оптимізації в іншому тощо).

Умовами прийняття готової версії продукту є:

– виконання всіх технічних вимог: продукт повинен повністю відповідати всім зазначеним технічним вимогам;

– відповідність вимогам стосовно інтерфейсу: інтерфейс користувача повинен бути простим, логічно структурованим та відповідати вимогам щодо групування функціоналу;

– тестування та верифікація: продукт повинен пройти повний цикл тестування;

– документація: повинна бути надана детальна документація, яка включає інструкції з користування, опис функціоналу.

Висновки до розділу 1

В ході написання першого розділу було розглянуто предметну область. Визначено фактори, що впливають на транспортні вантажоперевезення та їх планування. Розглянуто проблеми з оптимізацією перевезень у світі та на території Україні. Також було проведено аналіз аналогів. Кожен з них має свої переваги та недоліки. Спираючись на них було вказано вимоги до функціоналу та описано технічне завдання, за яким відбуватиметься розробка.

2 ПІДБІР МЕТОДІВ ДЛЯ ВИРІШЕННЯ ПОСТАВЛЕНОЇ ЗАДАЧІ

2.1 FuzzyTOPSIS

Алгоритм TOPSIS (Technique for Order Preference by Similarity to Ideal Solution) використовується для прийняття рішень у мультикритеріальних задачах, де альтернативи оцінюються за їхньою близькістю до ідеального розв'язку та віддаленістю від негативного ідеального розв'язку [14]. Проте у реальних ситуаціях часто зустрічаються нечіткість та неоднозначність, які важко врахувати за допомогою класичного TOPSIS.

FuzzyTOPSIS розширює цей підхід, використовуючи нечітку логіку для роботи з невизначеністю та нечіткістю в прийнятті рішень. Він дозволяє враховувати різні ступені достовірності та неоднозначності у вхідних даних, що робить його більш придатним для реальних задач, де точні числові дані можуть бути відсутніми або неповними [15].

Нижче наведено основні етапи алгоритму Fuzzy TOPSIS [14, 15].

1. Формування нечіткої матриці рішень: подібно до класичного TOPSIS, спочатку формується матриця рішень. Однак, замість точних числових значень, використовуються нечіткі числа (зазвичай трикутні або трапецієподібні нечіткі числа).

$$\tilde{X} = \begin{bmatrix} \tilde{x}_{11} & \tilde{x}_{12} & \cdots & \tilde{x}_{1n} \\ \tilde{x}_{21} & \tilde{x}_{22} & & \tilde{x}_{1n} \\ \vdots & & \ddots & \vdots \\ \tilde{x}_{m1} & \tilde{x}_{m2} & \cdots & \tilde{x}_{mn} \end{bmatrix},$$

де $\tilde{X}$ – нечітка матриця рішень;

$\tilde{x}_{ij}$ – значення j-го критерію для і-ї альтернативи;

m – кількість альтернатив;

n – кількість критеріїв.

2. Перетворення нечітких чисел у нечіткі нормалізовані числа: нечіткі числа нормалізуються за формулами, що враховують як критерії вигоди, так і критерії витрат.

Для критеріїв вигоди (великі значення кращі):

$$\tilde{r}_{ij} = \left(\frac{\tilde{x}_{ij} - \min\limits_i \tilde{x}_{ij}}{\max\limits_i \tilde{x}_{ij} - \min\limits_i \tilde{x}_{ij}} \right),$$

Для критеріїв витрат (менші значення кращі):

$$\tilde{r}_{ij} = \left(\frac{\max\limits_i \tilde{x}_{ij} - \tilde{x}_{ij}}{\max\limits_i \tilde{x}_{ij} - \min\limits_i \tilde{x}_{ij}} \right),$$

де $\tilde{r}_{ij}$ – нормалізоване значення j-го критерію для і-ї альтернативи;

$\min\limits_i \tilde{x}_{ij}$ – мінімальне значення j-го критерію серед усіх альтернатив;

$\max\limits_i \tilde{x}_{ij}$ – максимальне значення j-го критерію серед усіх альтернатив.

3. Визначення зваженої нормалізованої нечіткої матриці: кожен нормалізований нечіткий елемент множиться на відповідну вагу критерію w_j, яка також може бути нечітким числом.

$$\tilde{v}_{ij} = w_j * \tilde{r}_{ij},$$

де $\tilde{v}_{ij}$ – зважене нормалізоване значення;

w_j – вага j-го критерію.

4. Визначення нечіткого ідеального та антиідеального розв'язання: ідеальне розв'язання $\tilde{A}^+$ складається з найкращих значень нормалізованої матриці по кожному критерію, тоді як антиідеальне $\tilde{A}^-$ – з найгірших значень.

$$\tilde{A}^+ = \{\tilde{v}_1^+, \tilde{v}_2^+, \dots, \tilde{v}_n^+\} = \left\{\max_i \tilde{v}_{ij}\right\},$$

$$\tilde{A}^- = \{\tilde{v}_1^-, \tilde{v}_2^-, \dots, \tilde{v}_n^-\} = \left\{\min_i \tilde{v}_{ij}\right\},$$

де $\tilde{A}^+$ – ідеальне розв'язання (позитивний ідеал);

$\tilde{A}^-$ – антиідеальне розв'язання (негативний ідеал);

$\tilde{v}_j^+$ – зважене нормалізоване нечітке значення ідеального рішення для j-го критерію;

$\tilde{v}_j^-$ – зважене нормалізоване нечітке значення антиідеального рішення для j-го критерію.

5. Розрахунок відстаней до нечіткого ідеального та антиідеального рішень: використовується відстань між нечіткими числами (найчастіше Евклідова відстань або інші метричні відстані) для обчислення відстаней кожної альтернативи до ідеального D_i^+ та антиідеального D_i^- рішень.

$$D_i^+ = \sqrt{\sum_{j=1}^{n} \left(\tilde{v}_{ij} - \tilde{v}_j^+\right)^2},$$

де D_i^+ – відстань і-ї альтернативи до ідеального розв'язання $\tilde{A}^+$.

$$D_i^- = \sqrt{\sum_{j=1}^{n} \left(\tilde{v}_{ij} - \tilde{v}_j^-\right)^2},$$

де D_i^- – відстань і-ї альтернативи до антиідеального розв'язання $\tilde{A}^-$.

6. Обчислення нечіткої відносної близькості до ідеального розв'язання: відносна близькість альтернативи до ідеального рішення визначається як:

$$\tilde{C}_i^* = \frac{D_i^-}{D_i^+ + D_i^-},$$

де $\tilde{C}_i^*$ – показник відносної близькості і-ї альтернативи до ідеального розв'язання.

7. Ранжування альтернатив: альтернативи ранжуються на основі значень $\tilde{C}_i^*$. Найкраща альтернатива – та, яка має найбільше значення $\tilde{C}_i^*$.

Таким чином, алгоритм Fuzzy TOPSIS дозволяє більш адекватно враховувати невизначеність та нечіткість у даних, що робить його більш ефективним у багатьох реальних сценаріях, де дані є нечіткими або неповними.

2.2 Метод північно-західного кута

Метод північно-західного кута (NorthWestCorner method) є одним з найпростіших методів вирішення задачі транспортної логістики. Цей метод

використовується для знаходження початкового базисного розв'язку в задачі транспортних перевезень, де потрібно мінімізувати вартість перевезення товарів від кількох постачальників до кількох споживачів.

Транспортна задача зазвичай представлена у вигляді таблиці, де рядки відповідають постачальникам (з їх запасами), а стовпці - споживачам (з їх потребами). Кожна комірка таблиці представляє вартість перевезення одиниці товару від певного постачальника до певного споживача.

Починати заповнення таблиці потрібно з верхнього лівого (північно-західного) кута. Вибирається мінімальне значення між запасом першого постачальника і потребою першого споживача. Це значення розміщується в комірку (табл. 2.1). Запас і потреба відповідно зменшуються на це значення [16].

Таблиця 2.1 Побудова опорного плану методом північно-західного кута

Замовники / Відправники	30	10	15
25	25		
20			
10			

Після заповнення першої комірки [16]:

– якщо запас постачальника вичерпано (тобто дорівнює нулю), то рухаємось вниз до наступного постачальника;

– якщо потреба споживача вичерпана (тобто дорівнює нулю), то рухаємось праворуч до наступного споживача;

– якщо одночасно вичерпано і запас, і потребу, то можливі два варіанти руху: вниз або вправо. Обирається будь-який варіант (як правило, вправо).

Процес заповнення комірок повторюється, поки не буде заповнено всі комірки таблиці, що відповідають обмеженням на запаси і потреби (табл. 2.2).

Таблиця 2.2 Побудова опорного плану методом північно-західного кута

Замовники / Відправники	30	10	15
25	25		
20	5	10	5
10			10

Таким чином, було отримано початковий базисний розв'язок для задачі транспортної логістики за допомогою методу північно-західного кута. За бажанням, даний план може бути оптимізований, використовуючи відповідні методи.

2.3 Метод Фогеля

Метод Фогеля, який також відомий як метод двох мінімумів, наближений до оптимального плану, використовується для розв'язання транспортної задачі, яка полягає в оптимізації витрат на транспортування товарів з місця виробництва до місця споживання. Основними перевагами даного методу є те, що він простий в реалізації та має чітку зрозумілість процесу; метод забезпечує добрий початковий план, який часто близький до оптимального. Застосування цього методу до транспортних задач з нечіткими витратами відбувається, коли точні витрати не відомі, або коли вони можуть змінюватися в залежності від різних факторів.

Основна ідея методу Фогеля полягає в тому, щоб знайти початковий розв'язок задачі, розподіливши товари від місць виробництва до місць споживання з урахуванням найменшої вартості перевезення [17].

Щоб побудувати опорний план, який буде наближеним до оптимального необхідно для кожного рядка та стовбця вирахувати різницю двох найменших чисел, наприклад.

Таблиця 2.3 Побудова опорного плану (1 крок)

Замовники / Відправники	30	10	15	Штрафи
25	4	2	5	$\|4-2\|=2$
30	6	3	1	$\|3-1\|=2$
Штрафи	$\|4-6\|=2$	$\|2-3\|=1$	$\|5-1\|=4$	

Після чого, серед обчислених штрафів слід обрати той, що має найбільше число (різницю), в випадку показаний в табл. 2.3 це число 4. Далі обирається стовпчик, якому відповідає дана різниця, і в ньому береться мінімальне число, в даному випадку це число 1 третього стовпчика. Якщо найбільша різниця знаходиться в рядку, то пошук відбувається по ньому. Також, коли штрафи рівні між собою – враховуються значення стовпчиків [17]. Отже, знайшовши мінімальне значення у рядку/стовпчику, який відповідає максимальному штрафу, на його місце вписується максимальне значення для вантажу, що відправляється (табл 2.4).

Таблиця 2.4 Побудова опорного плану (2 крок)

Замовники / Відправники	30	10	15	Штрафи
25	4	2	5	$\|4-2\|=2$
30	6	3	15	$\|6-3\|=3$
Штрафи	$\|4-6\|=2$	$\|2-3\|=1$		

І в випадку коли замовник отримує ту кількість товару, яку очікує – даний рядок/стовпець виключається з побудови опорного плану. Розрахунки продовжуються, але для нової матриці витрат потрібно знову обрахувати штрафи [17]. Процес повторюється, поки всі товари не будуть розподілені.

Таблиця 2.5 Побудова опорного плану (3 крок)

Замовники / Відправники	30	10	15	Штрафи
25	4	2	5	
30	6	10	15	
Штрафи	$\|4-6\|=2$			

Коли залишиться один рядок/стовпчик (табл. 2.5), він заповнюється різницями того, що потрібно доставити:

$$30-10-15=5;$$

$$30-5=25.$$

В результаті всіх операції буде отримано наступний опорний план.

Таблиця 2.6 Опорний план за методом Фогеля

Замовники / Відправники	30	10	15
25	25	0	0
30	5	10	15

Сума витрат для даного плану становить:

$$S=100+0+0+30+30+15=175,$$

де S – сума витрат.

Скоріше всього, даний план не є оптимальним, через це - потрібно застосувати методи для подальшої оптимізації транспортних задач, але бувають випадки, коли метод Фогеля часто є оптимальним планом перевезень.

В даному прикладі (табл. 2.4 – 2.6), застосовується метод Фогеля для класичної матриці витрат. Однак, щоб використати його для нечітких – потрібно модифікувати наступним чином [17]:

– нечіткі витрати: витрати транспортування представлені у вигляді нечітких чисел (наприклад, трикутних або трапецієподібних);

– обчислення нечітких різниць витрат: для кожного рядка та стовпця обчислюють нечіткі різниці витрат аналогічно класичному методу, але з використанням операцій над нечіткими числами. В цій роботі буде застосовано наступний обрахунок різниць - $(a + b + c)/3$, де a,b,c – нечіткі числа;

– вибір комірки з нечіткими витратами: обирають комірку з мінімальними нечіткими витратами на основі визначених пріоритетів;

– розподіл на нечітких основах: виконується розподіл товарів, враховуючи нечіткі значення попиту та пропозиції;

– коригування таблиці: застосовується корекція для нечітких попиту та пропозиції.

Метод Фогеля для нечітких витрат дозволяє ефективно працювати з невизначеністю в задачах оптимізації транспортних витрат, зберігаючи при цьому переваги класичного підходу. Реалізація даного методу на мові програмування C# наведена в додатку Б.

2.4 Метод потенціалів

Метод потенціалів - це ефективний і гнучкий метод для розв'язання транспортних задач. Він базується на ідеї використання потенціалів для знаходження оптимальних маршрутів в мережі. Основна мета методу полягає в мінімізації загального витрат або часу, пов'язаних з переміщенням ресурсів від відправника до замовника. Його гнучкість та можливість використання у поєднанні з різними алгоритмами пошуку оптимальних шляхів роблять його популярним в наукових дослідженнях та практичних застосуваннях.

Перевагами даного методу є [18]:

– точність і надійність: метод потенціалів гарантує знаходження оптимального рішення за умови правильно побудованого початкового плану; Тобто,

при його використанні можна бути впевненим у мінімізації загальних витрат на перевезення;

– математична обґрунтованість: метод базується на чітких математичних принципах і теоріях (теорія потенціалів і двоїстості), що забезпечує його надійність і передбачуваність;

– ефективність обчислень: дозволяє значно скоротити кількість обчислень порівняно з іншими методами, такими як повний перебір. Це досягається за рахунок зменшення кількості розглянутих варіантів перевезень через використання потенціалів;

– можливість використання для великих задач: даний метод добре масштабується для великих задач транспортування. Хоч кількість обчислень зростає з розміром задачі, метод залишається відносно ефективним завдяки систематичному підходу до оптимізації;

– гнучкість і адаптивність: може бути адаптований до різних варіантів транспортних задач, включаючи задачі з додатковими умовами або обмеженнями, такими як обмеження на максимальний вантаж на одній лінії, чи нечіткими витратами;

– інтеграція з іншими методами: метод потенціалів можна ефективно комбінувати з іншими методами оптимізації. Наприклад, він може бути використаний як частина більшого алгоритму для подальшого уточнення рішення, отриманого іншими методами;

– покращення початкових планів: він особливо корисний для покращення початкових планів перевезень, які можуть бути отримані іншими методами, такими як метод Фогеля, північно-західного кута або метод мінімальних витрат.

Недоліками, в свою чергу, є:

– складність початкового розуміння: для новачків даний метод може здатися складним через необхідність глибокого розуміння теорії потенціалів, двоїстості та побудови циклів;

– обчислювальна складність: хоча метод потенціалів ефективний для великомасштабних задач, кількість обчислень зростає з розміром задачі. Це може вимагати значних обчислювальних ресурсів і часу, особливо для дуже великих задач або задач з великою кількістю обмежень;

– побудова циклів: процес побудови циклів для коригування плану може бути складним і трудомістким, особливо при великій кількості пунктів відправлення та призначення. Виявлення і правильне формування циклів вимагає уваги до деталей і точних обчислень.

В першу чергу, для розв'язання транспортної задачі необхідно ініціалізувати потенціали тих рядків та стовпчиків, де вказана кількість товару для транспортування (входять в рішення транспортної задачі) [18]. Приклад наведено для таблиці 2.6.

Таблиця 2.7 Знаходження потенціалів рядків та стовпців

Замовники / Відправники	4 30	1 10	-1 15
0 25	4 25	2	5
2 30	6 5	3 10	1 15

Потенціал першого рядка завжди дорівнює 0, звідси, за формулою 2.1 можна розрахувати для кожного рядка та стовпця:

$$U_i + V_j = C_{ij}, \tag{2.1}$$

де U_i – потенціал рядка;

V_j – потенціал стовпця;

C_{ij} – витрати.

Тобто, $U_1 = \mathbf{0}, V_1 = 0 + \mathbf{4} = 4, U_2 = \mathbf{2} + 4 = 6, V_2 = 2 + \mathbf{1} = 3,$

$V_3 = 2 + (-\mathbf{1}) = 1.$

Після того, як знайдено потенціали рядків та стовпців, необхідно знайти потенціали тих комірок, які в рішення не входять (в даному випадку це C_{12} та C_{13}) [18]. Виконати це можна за наступною формулою 2.2:

$$\Delta_{ij} = C_{ij} - U_i - V_j, \quad (2.2)$$

де Δ_{ij} – потенціал помірки, що не входить в рішення.

Звідси, $\Delta_{12} = 2 - 0 - 1 = 1$, $\Delta_{13} = 5 - 0 - 1 = 4$. Дивлячись на отримані результати, якщо серед комірок, що не входять в рішення немає від'ємних потенціалів – даний план транспортних перевезень являється оптимальним. Тобто, застосувавши метод Фогеля було отримано оптимальний план.

Якщо, після виконання дій, серед комірок, що не входять в рішення наявне хоча б одне від'ємне число – той план не є оптимальних і оптимізацію слід продовжити.

Таблиця 2.8 Інший план транспортних вантажоперевезень

Замовники / Відправники	4	2	-1
	30	10	15
0	4	2	5
25	15	10	
2	6	3	1
30	15		15

Для випадку, наведеного в таблиці 2.8 $U_1 = \mathbf{0}, V_1 = 0 + \mathbf{4} = 4$,
$V_2 = 0 + \mathbf{2} = 2, U_2 = \mathbf{2} + 4 = 6, V_3 = 2 + (-\mathbf{1}) = 1$.

Інші комірки: $\Delta_{13} = 5 - 0 - (-1) = 6$, $\Delta_{22} = 3 - 2 - 2 = -1$. В даному випадку, комірка C_{22} має потенціал -1, це вказує на те, що план (табл. 2.8) не є оптимальним і його слід оптимізувати. Якщо комірок з від'ємним потенціалом декілька – береться та, яка більше серед інших. Коли така комірка знайдена – створюється цикл перерозподілу транспортування [18]. Циклом являється замкнута

фігура, яка розташована на комірках які входять в рішення і складається виключно з прямих.

Таблиця 2.9 Побудування циклу задля оптимізації плану

Замовники / Відправники	4 30	2 10	-1 15
0 25	4 15	2 10	5
2 30	6 15	3	1 15

Для того, щоб комірка C_{22} входила в рішення, їй необхідно вказати необхідну кількість товару для перевезення. Щоб дізнатися, яку кількість потрібно додати – слід глянути на комірки, що знаходяться з боків, і обрати мінімальне значення, в даному випадку – комірка $C_{12} = 10$. Тепер необхідно додати та відняти значення комірки C_{12} для інших.

Таблиця 2.10 Додавання та віднімання мінімального значення для інших комірок по циклу

Замовники / Відправники	4 30	2 10	-1 15
0 25	4 + 15	2 - 10	5
2 30	6 - 15	3 +	1 15

Тобто, буде отримано наступне:

- для комірки $C_{12} = 10 - 10 = 0$;
- для $C_{11} = 15 + 10 = 25$;

– $C_{21} = 15 - 10 = 5$;

– $C_{22} = 0 + 10 = 10$.

Таблиця 2.11 Завершення розрахунку циклу

Замовники / Відправники	4	2	-1
	30	10	15
0	4	2	5
25	25	0	0
2	6	3	1
30	5	10	15

Дивлячись на план, отриманий в таблиці 2.11, можна побачити що він ідентичний плану таблиці 2.7 і тому є оптимальним.

В іншому випадку, необхідно знову підрахувати потенціали для рядків та стовпчиків, комірки яких входять в рішення. Потім підрахувати потенціали для комірок, які не входять в рішення, якщо серед цих комірок немає від'ємних потенціалів – той план перевезень є оптимальним. Якщо є – продовжувати оптимізацію зі створення циклу перерозподілу [18].

Як і для методу Фогеля, приклад було наведено для класичної матриці витрат. В свою чергу, для нечітких витрат знаходження потенціалів матиме іншу логіку. В даному випадку – $(a + b + c)/3$, де a,b,c – нечіткі числа.

Приклад реалізації методу потенціалів мовою програмування C# наведено в додатку В.

Висновки до розділу 2

У другому розділі було досліджено чотири методи розв'язання транспортних задач: два для побудови опорного плану (методи Фогеля та північно-західного кута), два для їх оптимізації (FuzzyTOPSIS та потенціалів). В подальшому виконанні роботи буде використано лише методи Фогеля та потенціалів. Загалом, дані два методи

відіграють важливу роль у розв'язанні транспортних задач. Алгоритм Фогеля швидко надає допустиме рішення, яке потім може бути покращено за допомогою методу потенціалів до досягнення оптимального розподілу. Метод потенціалів, в свою чергу, гарантує знаходження оптимального рішення за умови правильного початкового плану. Це означає, що при його використанні можна бути впевненим у мінімізації загальних витрат на перевезення. Практичні приклади, наведені у розділі, підтверджують ефективність і застосовність цих методів у реальних ситуаціях, що робить їх незамінними інструментами для фахівців з оптимізації логістичних процесів.

3 РЕАЛІЗАЦІЯ ПРОГРАМНОГО ЗАСТОСУНКУ

3.1 Обґрунтування вибору технологій розробки

Для розробки застосунку буде використано інтерфейс програмування додатків Windows Forms. Це один з інструментів для створення графічних користувацьких інтерфейсів (GUI) в програмах для операційної системи Windows.

Windows Forms надає простий та інтуїтивно зрозумілий спосіб створення інтерфейсу користувача за допомогою графічного дизайнера Visual Studio (рис. 3.1) або іншої інтегрованого середовища розробки (IDE) [19].

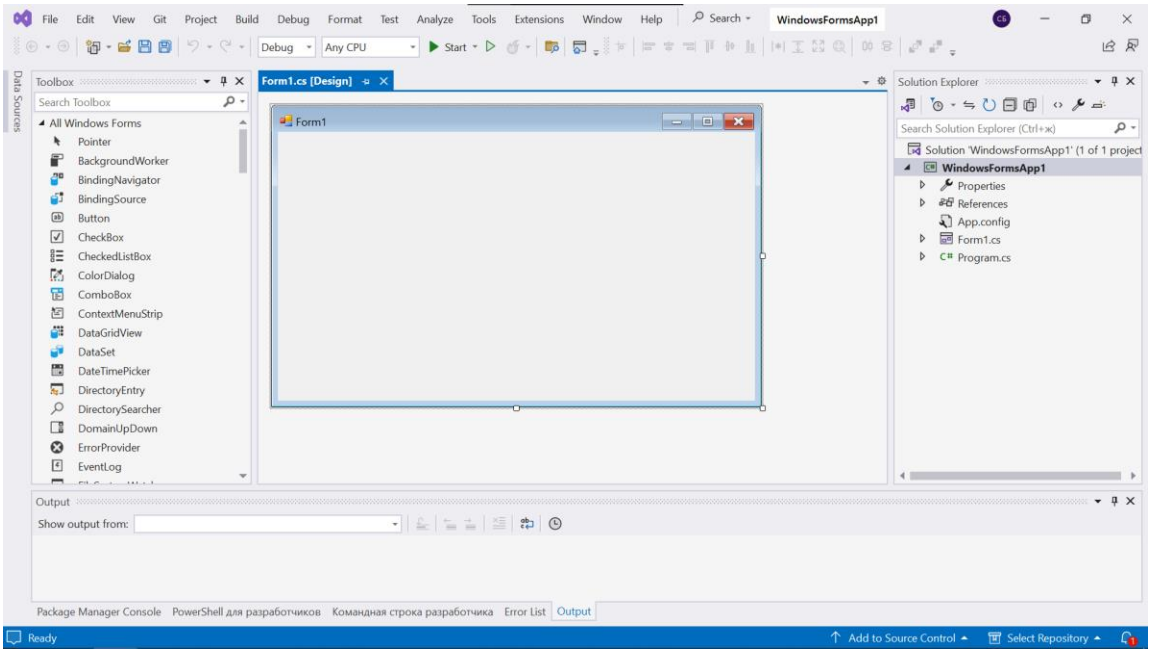

Рисунок 3.1 – Інтерфейс середовища розробки Visual Studio

Завдяки простоті використання і готовому набору елементів інтерфейсу, розробка програм на Windows Forms зазвичай відбувається швидше, ніж з використанням інших технологій. А базування Windows Forms на .NET Framework робить його частиною великої кількості інструментів та бібліотек, що дозволяє легко інтегрувати з іншими компонентами програмної системи.

Дивлячись на рис. 3.1 можна побачити, що середовище розробки містить такі вікна, як Toolbox, Windows Form Designer, Solution Explorer.

Коротко про дані вікна:

– Toolbox містить колекцію елементів керування і компонентів, які можна використовувати для створення інтерфейсів користувача. Наприклад, Button - створює кнопку, на яку користувач може натискати, Label - відображає текст або зображення, TextBox - дозволяє користувачеві вводити текст та безліч інших [20];

– Windows Form Designer в Visual Studio надає візуальне середовище, в якому розробники можуть легко створювати та налаштовувати форми, додаючи на них різні елементи керування через інтерфейс "drag-and-drop";

– Solution Explorer - це ключовий інструмент в Visual Studio, який допомагає розробникам організовувати, переглядати та керувати файлами і проектами у рішенні (solution). Він надає зручний спосіб навігації по всіх компонентах вашого проекту або рішення, дозволяючи вам швидко знаходити і відкривати файли, налаштовувати властивості проектів, додавати нові файли та багато іншого [21].

Також, за допомогою меню View можна відобразити більше вікон, необхідних для розробки, наприклад Properties Window. Даний інструмент є одним з ключових в Visual Studio. Він дозволяє розробникам переглядати та змінювати властивості елементів керування та об'єктів у середовищі розробки. Вікно властивостей відображає детальну інформацію про вибраний об'єкт, дозволяючи налаштовувати його без необхідності писати код вручну.

3.2 Опис компонентів застосунку

При створенні нового застосунку Windows Forms, разом з ним створюються декілька компонентів, які керують різними аспектами його роботи, а саме Properties, References, App.config, Form1.cs та Program.cs.

3.2.1 Properties

Вузол Properties містить файли властивостей проекту, які визначають його конфігурацію та налаштування (рис. 3.2). Ці файли, зазвичай project.properties та AssemblyInfo.cs, містять інформацію про:

– назву, точку входу, версію проєкту, тощо;

– метадані: додаткова інформація про проект, така як опис, авторські права та ліцензія;

– налаштування складання: параметри, що керують процесом складання проекту, такі як цільова платформа, тип вихідного файлу та символи умовного компіляції;

– налаштування розгортання: параметри, що визначають, як проект буде розгорнуто, наприклад, шлях до каталогу призначення та файли конфігурації;

– параметри документації: налаштування для генерації документації для вашого проекту, наприклад, формат документації та рівень деталізації;

– інші налаштування: додаткові конфігурації, що впливають на поведінку проекту.

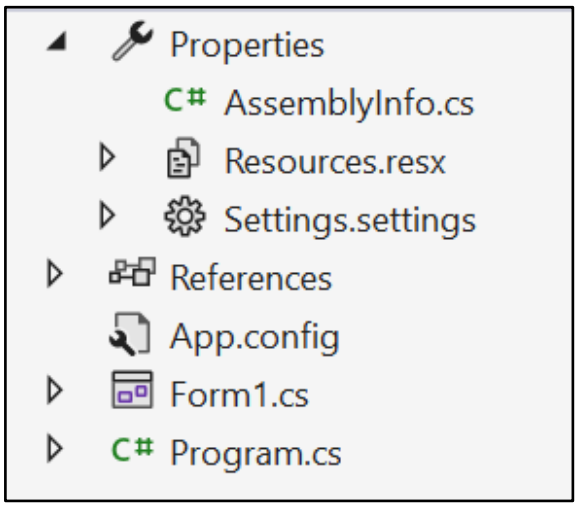

Рисунок 3.2 – Вміст компоненту Properties

3.2.2 References

Цей розділ містить посилання на зовнішні бібліотеки та збірки, які використовує даний проект (рис. 3.3). Сюди відносяться системні бібліотеки (System, System.Data, System.Windows.Forms), користувацькі бібліотеки (створені іншими розробниками, які додаються вручну), пакети NuGet (зовнішні пакети, можна додати через менеджер пакетів NuGet) [22].

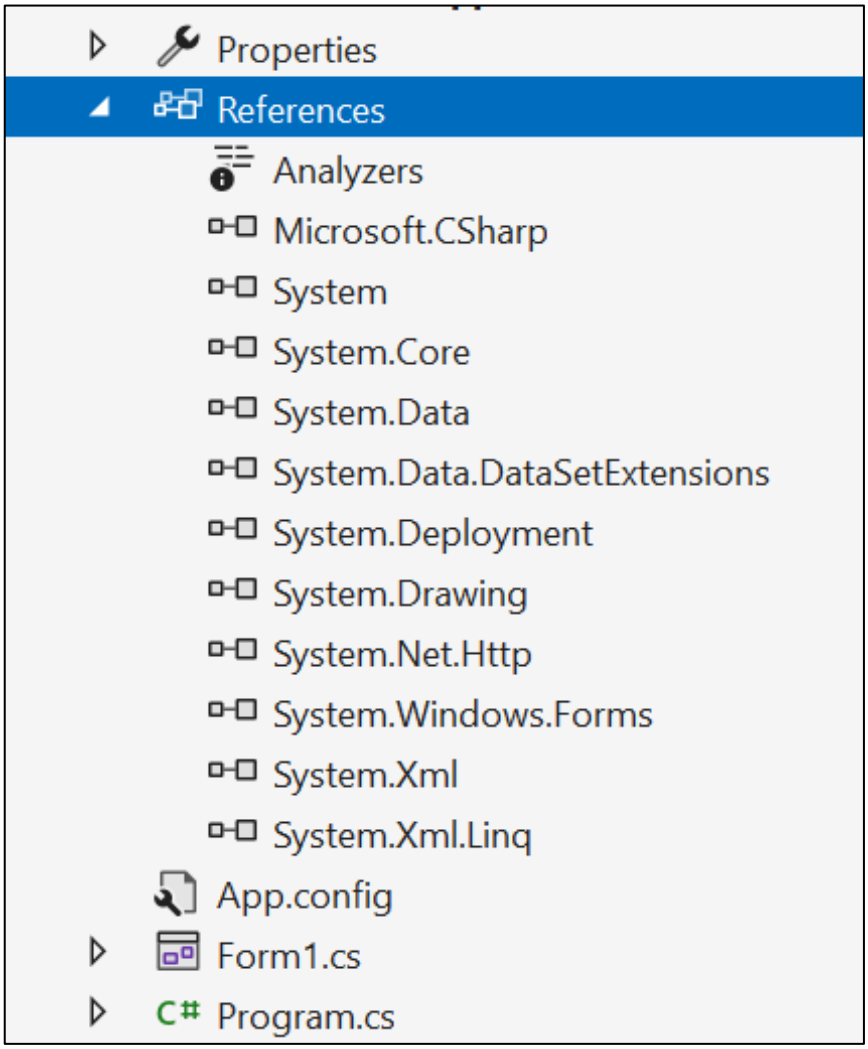

Рисунок 3.3 - Вміст компоненту References

Analyzers містить аналізатори коду, які допомагають виявляти помилки, потенційні проблеми та пропонувати покращення для коду під час його написання.

Microsoft.CSharp надає служби для компіляції та виконання коду C#. Вона необхідна для роботи з динамічними типами і виконання C# коду на льоту.

System це основна бібліотека .NET Framework, яка містить базові типи і служби, такі як колекції, введення/виведення, строки, час, математика тощо.

System.Core – збірка містить основні типи і функціональність для LINQ (Language Integrated Query), а також розширювальні методи для основних типів .NET.

System.Data – бібліотека для роботи з базами даних, містить класи для доступу та управління даними з різних джерел даних, таких як SQL Server.

System.Data.DataSetExtensions надає розширення для роботи з DataSet і DataTable, включаючи підтримку LINQ to DataSet.

System.Deployment – збірка для підтримки ClickOnce деплойменту, що дозволяє розгортати застосунки з мінімальними зусиллями.

System.Drawing це бібліотека для роботи з графікою, забезпечує класи для малювання тексту, графіки та зображень.

Бібліотека System.Net.Http містить класи для роботи з HTTP протоколом, включаючи надсилання HTTP-запитів і отримання HTTP-відповідей.

System.Windows.Forms – основна бібліотека для створення Windows Forms застосунків. Містить всі необхідні класи для створення графічного інтерфейсу користувача.

System.Xml – збірка для роботи з XML-документами, включає класи для парсингу, зчитування, створення та зміни XML-документів.

System.Xml.Linq надає можливості для роботи з XML, використовуючи LINQ (Language Integrated Query). Містить класи для роботи з XML в стилі LINQ.

3.2.3 App.config

Цей файл містить конфігураційні налаштування застосунку. Він часто використовується для зберігання налаштувань, які можуть змінюватися без необхідності перекомпіляції коду. Наприклад, строки підключення до баз даних

(Connection Strings); загальні налаштування, які можуть бути зчитані в коді (App Settings); Спеціалізовані налаштування для конкретних бібліотек або служб (Configuration Sections).

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
        <startup>
                <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.7.2" />
        </startup>
```

Вище наведено вміст файлу App.config, де `<?xml version="1.0" encoding="utf-8" ?>` – декларація XML-версії та кодування. Вказує, що цей файл є XML-документом версії 1.0 та використовує кодування UTF-8. `<configuration>` – тег визначає початок конфігураційного файлу. `<startup>` – вказує на початок секції налаштувань запуску. `<supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.7.2" />` – тег, що визначає підтримувану версію .NET Framework для запуску додатку. В даному випадку, додаток підтримує версію .NET Framework 4.7.2. Отже, цей конфігураційний файл вказує, що додаток повинен бути запущений на .NET Framework версії 4.7.2 або вище.

3.2.4 Файли Form1.cs та Program.cs

Файл Form1.cs містить код для головної форми застосунку. Це включає визначення і поведінку елементів управління, таких як кнопки, текстові поля, меню тощо. Відкривши його, у візуальному дизайнері Visual Studio, можна розміщувати та налаштовувати елементи управління на формі. Код для цих елементів управління генерується автоматично і зберігається в Form1.Designer.cs.

Основний файл Form1.cs містить код, написаний для обробки подій, таких як натискання кнопок або введення тексту. Наприклад, можна додати методи для обробки подій Click або TextChanged, що пов'язані з відповідними елементами управління. Нижче наведено приклад застосування.

```
public partial class Form1 : Form
{
	public Form1()
	{
		InitializeComponent();
	}
	private void button1_Click(object sender, EventArgs e)
	{
		MessageBox.Show("Button clicked!");
	}
}
```

В конструкторі даного класу викликається метод InitializeComponent(), який відповідає за ініціалізацію всіх компонентів форми. Цей метод автоматично генерується Visual Studio і міститься у файлі Form1.Designer.cs. Далі оголошено метод, який є обробником події, що виконується, коли користувач натискає кнопку button1.

Стосовно файлу Program.cs, він містить головну точку входу програми. Саме тут програма починає своє виконання.

Метод Main зазвичай статичний (static) і є точкою входу, звідки починається виконання програми. В ньому встановлюються налаштування для запуску форми, такі як встановлення культури для локалізації, обробка винятків тощо.

```
static class Program
{
   [STAThread]
   static void Main()
   {
      Application.EnableVisualStyles();
      Application.SetCompatibleTextRenderingDefault(false);
      Application.Run(new Form1());
   }
}
```

У методі Main створюється екземпляр головної форми (Form1) і запускається додаток за допомогою Application.Run.

Ці два файли (Form1.cs та Program.cs) працюють разом, щоб забезпечити запуск і роботу програми Windows Forms. Program.cs відповідає за старт програми, тоді як Form1.cs керує графічним інтерфейсом та логікою взаємодії з користувачем.

3.3 Проектування користувацького інтерфейсу

Опираючись на вимоги, описані в 1 розділі, користувацький інтерфейс повинен бути простим для розуміння. Користувач, дивлячись на компоненти застосунку, на інтуїтивному рівні повинен розуміти, який функціонал він виконує.

Перш ніж будувати початковий план перевезень, користувач повинен мати змогу ввести дані, які міститимуть інформацію про відправників, замовників та кількість товару, яку необхідно транспортувати.

Для цього, на формі, було створено об’єкт Panel, який містить в собі об’єкти Label, TextBox і Button (рис. 3.4).

Рисунок 3.4 – Панель вводу даних користувачем

В текстові поля Supply та Demand потрібно буде вводити кількість товару, яку потребує замовник від постачальника.

Наприклад, ввівши для Supply (30, 20, 25), а для Demand (40, 15, 20) буде отримано план як показано в таблиці 3.1.

Таблиця 3.1 – План перевезень вантажу

	40	15	20
30			
20			

25			

Після на натискання на кнопку Input Costs, у користувача з'явиться нове вікно – для вводу витрат на перевезення (рис. 3.5).

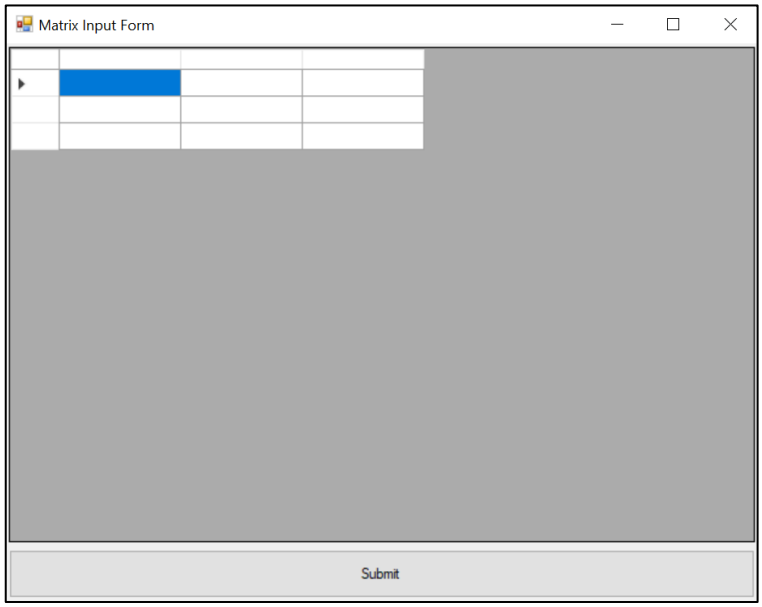

Рисунок 3.5 – Вікно для вводу витрат на перевезення

Розмірність даної таблиці вираховується з кількості відправників та замовників, тому користувач зможе заповнити дані витрати до перевезень не помилившись при їх введенні.

Останнє, що залишилось додати, це панель для виводу результатів планування, та кнопки для виведення даних користувача, початку роботи методу потенціалів та збереження отриманих результатів.

В якості об’єкту для виведення на нього отриманих результатів було взято Textbox. Одним з головним його плюсів є підтримка операції копіювання, вирізання і вставки тексту, що підвищує зручність користування. Також Textbox легко інтегрується з іншими елементами управління, такими як кнопки, списки, панелі та інші, що дозволяє створювати багатофункціональні форми з комплексною логікою роботи.

Єдине, що залишається – це додати кнопки для початку обрахунків та збереження результатів в .txt файл, щоб відповідати вимогам технічного завдання, описаного в 1 розділі. Ні чим особливим дані кнопки зараз не відрізняються, лише в майбутньому – матимуть різний функціонал.

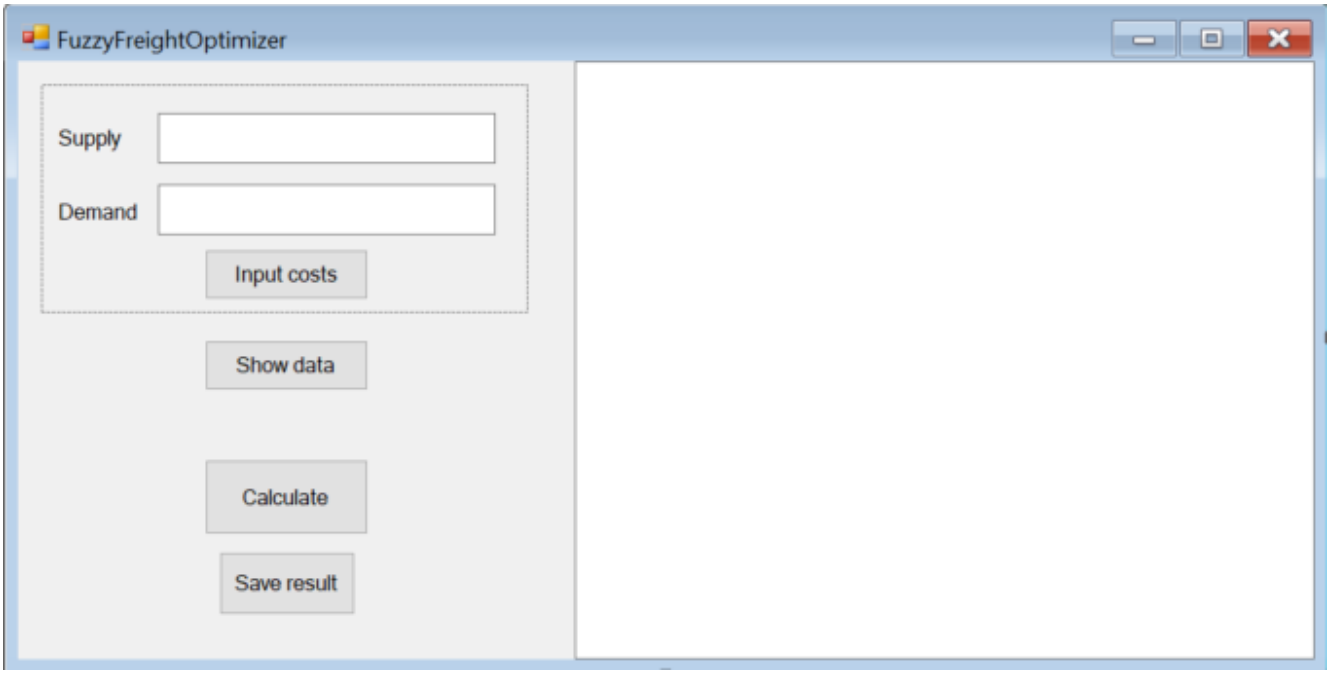

Рисунок 3.6 – Кінцевий вигляд застосунку FuzzyFreightOptimizer

Маючи готовий користувацький інтерфейс (рис. 3.6), можна переходити до програмування його компонентів, обрахунку плану, застосування методу Фогеля та потенціалів тощо.

3.4 Програмування основної логіки застосунку

3.4.1 Отримання даних користувача

Спершу потрібно, щоб при натисканні кнопки Input costs відбувалась перевірка чи було введено дані, якщо ні – повідомити користувача.

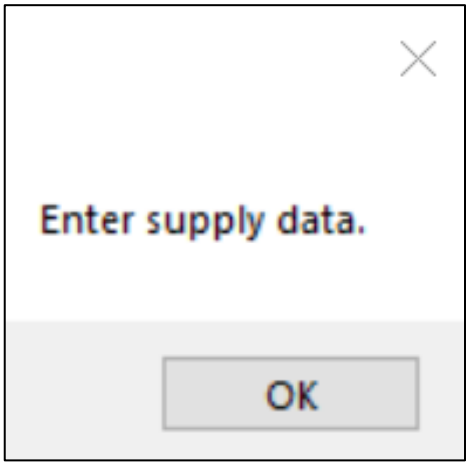

Рисунок 3.7 – Повідомлення про введення даних для поля Supply

Якщо користувач, все ж таки, не ввів дані про відправників та замовників він отримає такі повідомлення (рис. 3.7, рис. 3.8).

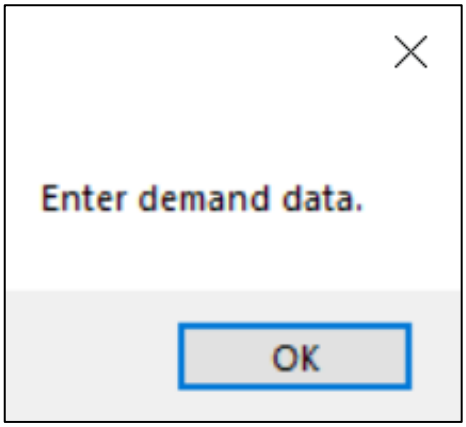

Рисунок 3.8 – Повідомлення про введення даних для поля Demand

Якщо дані введено, їх потрібно перевести з типу рядка (string) в масив (int[,]). Для цієї задачі було реалізовано метод ParseStringToIntArray. Спочатку, в ньому, відбувається розділення вхідного рядка на частини за комою та пробілом.

```
string[] parts = input.Split(new string[] { ", " }, StringSplitOptions.RemoveEmptyEntries);
```

Далі, з цього масиву, формується результат – в циклі, використовуючи метод int.Parse(), рядки масиву parts переводяться в цілісний тип. При виході з циклу, масив вже з числами повертається.

Після чого, користувач має змогу ввести матрицю витрат, для опорного плану. Відповідно для цього створюється форма Matrix Costs Input Form (рис. 3.9).

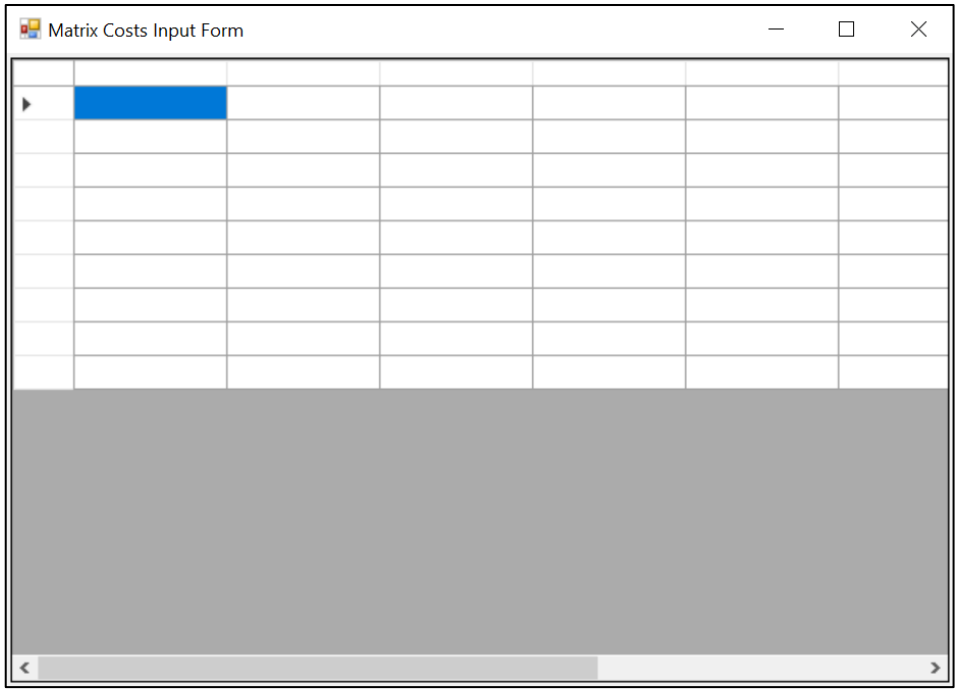

Рисунок 3.9 - Matrix Costs Input Form

Для отримання даних з DataGridView і подальшого їх передавання з цієї форми в іншу було написано наступний код. Ось як він працює:

– ініціалізація змінних: отримуються кількість рядків та стовпців у DataGridView;

– створення матриці FuzzyNumber: створюється двовимірний масив data, який буде містити числа типу FuzzyNumber. Розміри матриці відповідають кількості рядків та стовпців у DataGridView;

– заповнення матриці з DataGridView: проходиться по кожній комірці DataGridView, отримуються значення як рядок тексту. Потім рядок розділяється на трійку чисел, розділених комами. Ці числа конвертуються у відповідні значення типу double, які використовуються для створення нового об'єкта типу FuzzyNumber. Цей об'єкт зберігається у відповідній позиції матриці data;

– виклик події DataSubmitted: якщо подія DataSubmitted зареєстрована, вона викликається із зібраними даними. Це дозволяє іншим частинам програми реагувати на подію та обробляти отримані дані;

– закриття форми: після отримання та передачі даних форма, яка містить DataGridView, закривається.

Після того, як всі необхідні дані отримано, йде черга за обрахунками, але через те, що користувач міг ввести невірні данні програма може невірно спрацювати і, отриманий в результаті, план не буде оптимальним. Тому щоб запобігти цьому було зроблено виведення всіх даних в Textbox (рис. 3.10), щоб користувач міг їх переглянути і у випадку помилкових даних – виправити їх повторних введенням.

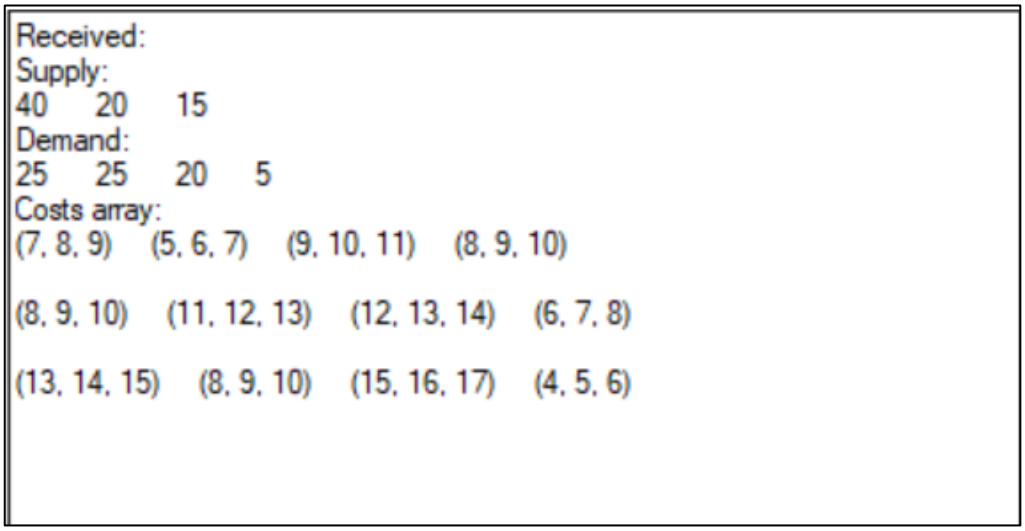

Рисунок 3.10 – Виведення користувацьких даних після натискання кнопки Showdata

Реалізація виглядить досить просто – циклом, у Textbox виводяться постачальні потреби (supply) та попит (demand). Вслід за ними, виводиться матриця витрат в теж саме поле.

У випадку, коли користувач введе невірні дані, у нього буде змога переконатись чи все вірно і якщо буде необхідність виправити їх.

3.4.2 Реалізація методу Фогеля

Маючи весь необхідний функціонал для отримання даних від користувача, можна переходити до реалізації самих методів оптимізації транспортних вантажоперевезень

Спочатку, маючи всі необхідні дані про перевезення ресурсів, можна побудувати початковий план перевезень застосувавши метод Фогеля (Додаток А). Основна ідея методу полягає в тому, щоб знайти клітинку з найбільшою вартістю в кожному рядку і стовпчику, порівняти їх та обрати клітинку з найбільшою різницею. Потім обчислюється обсяг перевезень для цієї клітинки, поки виконується умова завершення.

Нижче наведено ключові етапи роботи коду:

– ініціалізація змінних: створюються масиви rowDone та colDone для відстеження завершення обробки рядків та стовпців відповідно;

– обчислення штрафів: для кожного незавершеного рядка та стовпця обчислюється "пенальті" на основі двох найменших вартостей у рядку або стовпці. Ці значення зберігаються у відповідних масивах rowPenalties та colPenalties;

– вибір клітинки з максимальним штрафом: вибирається клітинка з максимальним штрафом серед усіх рядків та стовпців. Якщо максимальні штрафи однакові, обирається рядок або стовпчик з більшим штрафом;

– розподіл обсягу перевезень: обчислюється обсяг перевезень для вибраної клітинки, що дорівнює мінімуму між доступними запасами та потребами у відповідних рядку та стовпці. Результат зберігається у матриці result, а обсяги перевезень зменшуються відповідно;

– оновлення відстежувальних масивів: якщо запаси або потреби вичерпані для якогось рядка або стовпця, відповідні прапорці в масивах rowDone та colDone встановлюються в true.

Цей алгоритм продовжується до тих пір, поки всі запаси та потреби не будуть вичерпані.

3.4.3 Реалізація методу потенціалів

Розв’язання задачі транспортного планування з використанням модифікації методу Фогеля, а також методу потенціалів реалізованого наступним чином:

– ініціалізація змінних: отримуються розміри таблиці витрат і створюється масив result для зберігання розв'язку;

– виклик методу Фогеля для наближеного розв'язання: початковий наближений розв'язок отримується викликом методу VogelApproximation, який заповнює масив result згідно з методом Фогеля;

– обчислення потенціалів: обчислюються потенціали u та v, які допомагають вирішити нерівності між вартостями перевезення;

– виправлення розв'язку за допомогою методу потенціалів: цикл продовжується, поки є клітинки, які не заповнені (розв'язок не є оптимальним). Знаходиться клітина з мінімальною дельтою (різницею між вартістю перевезення та сумою потенціалів), і розв'язок коригується за допомогою методу AdjustSolution;

– повернення результату: після завершення всіх ітерацій повертається знайдений розв'язок.

Даний код використовує комбінацію методів Фогеля та потенціалів для ефективного вирішення задачі транспортного планування з нечіткими витратами. Повна його реалізація наведена в додатку В.

3.4.4 Вивід отриманих результатів користувачу

Після отримання даних та проведення всіх розрахунків з планування та оптимізації транспортних вантажоперевезень з нечіткими витратами, необхідно повідомити користувача про отримані результати. Для цього було написано код, що виводитиме в TextBox все те, що було описано вище.

При натисканні на кнопку Calculate, відбувається розв’язання транспортної задачі, після чого виводить оптимальний план та обчислює його загальні витрати. Виведення плану відбувається за рахунок виведення кожного значення матриці result у TextBox, в якості таблиці. Також, після кожного рядка додається дві порожні строки для розділення рядків.

Для даних, які вказано на рисунку 3.10 оптимальний план виглядатиме наступним чином (рис. 3.11).

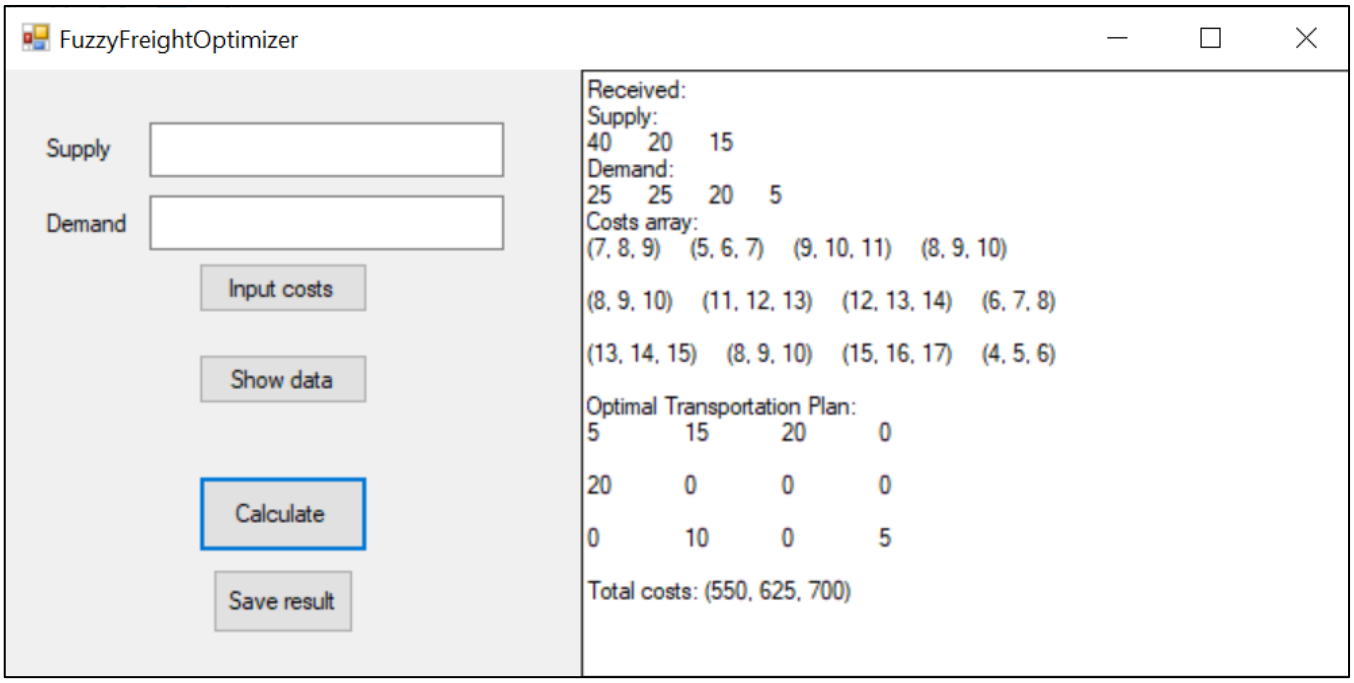

Рисунок 3.11 – Виведення результатів оптимізації транспортних перевезень

Окрім оптимального плану, користувач також отримає загальні витрати до нього (Total costs).

3.5 Тестування застосунку

В першу чергу, потрібно дізнатись як даний застосунок реагує на неправильно введені дані. Спочатку введемо від’ємні значення до supply та demand:

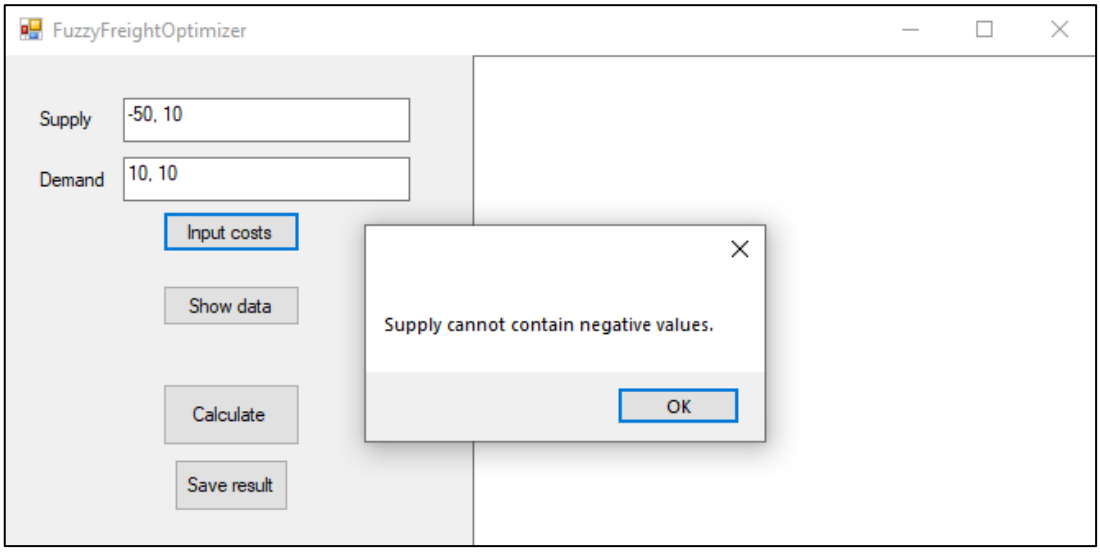

Рисунок 3.12 – Тестування застосунку на введення від'ємних значень

Дивлячись на рисунок 3.12 можна побачити, що при введені $-50, 10$ – програма відпрацювала правильно. Після чого, важливо перевірити, чи дорівнює сума елементів supply сумі demand (рис. 3.13).

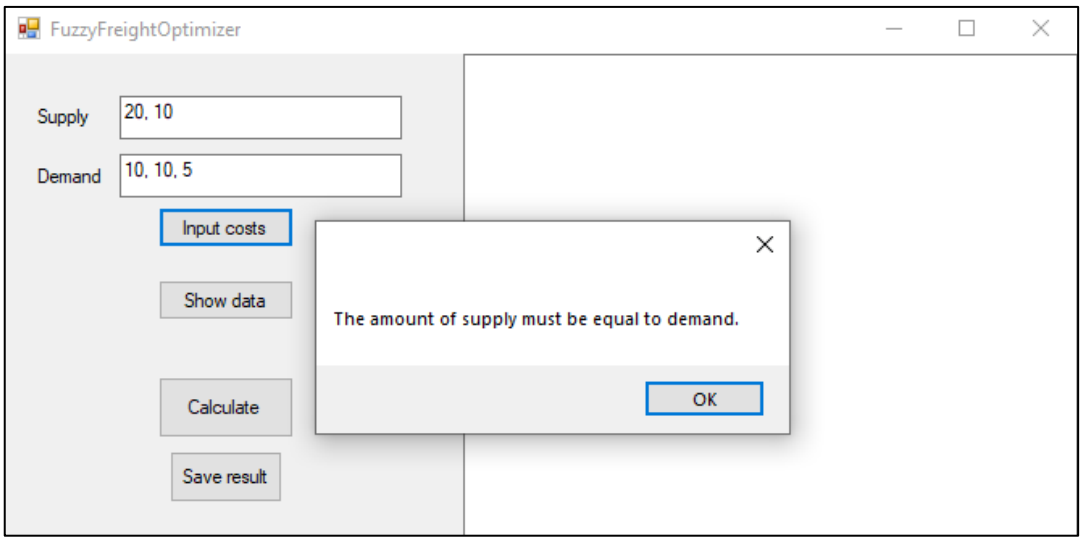

Рисунок 3.13 – Тестування застосунку на рівність сум елементів supply та demand

На даному етапі, користувача буде повідомлено про неправильно введені дані, стосовно supply та demand, якщо такі є.

Далі необхідно провести тестування правильність введення нечітких витрат. Правильними вважаються такі приклади: (1,2,3); (1, 2, 3); 1,2,3; 1, 2, 3. Також особливим є правило, що в трикутних нечітких витратах $a < b < c$, де (a, b, c). Воно перевіряється наступним кодом.

```
string pattern = @"^\(?(\d+),\s*(\d+),\s*(\d+)\)?$";
Match match = Regex.Match(input, pattern);

if (!match.Success)
{
    return false;
}

int a = int.Parse(match.Groups[1].Value);
int b = int.Parse(match.Groups[2].Value);
int c = int.Parse(match.Groups[3].Value);

return a < b && b < c;
```

І ввівши неправильні значення в форму нечітких витрат отримаємо результат як на рисунку 3.14. В іншому випадку, витрати буде зчитано і застосунок продовжить свою роботу.

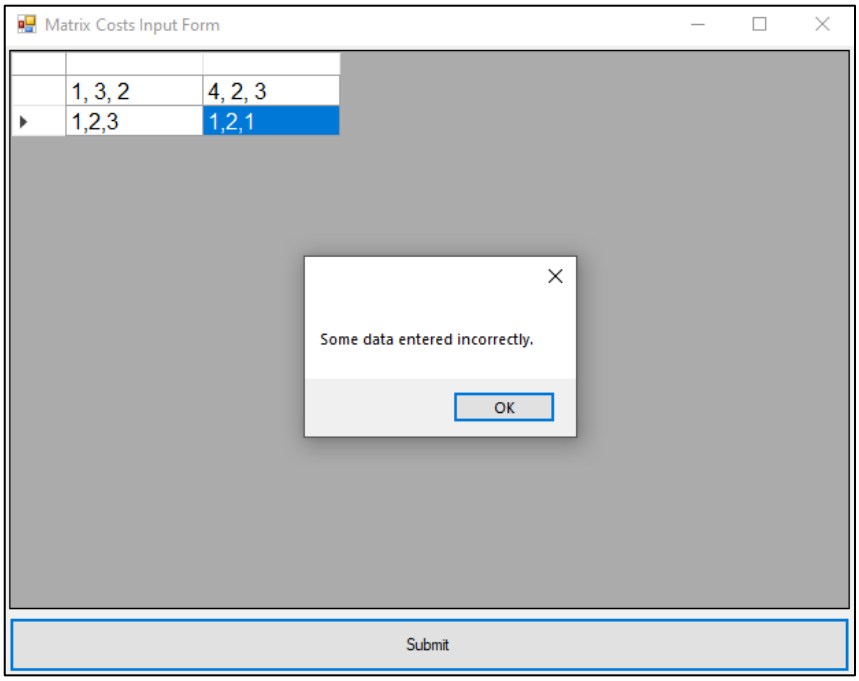

Рисунок 3.14 Тестування введення нечітких витрат

Єдине, що залишилось протестувати – роботу кнопок. Наприклад, як відреагує застосунок та яке отримає повідомлення, якщо даних не було введено.

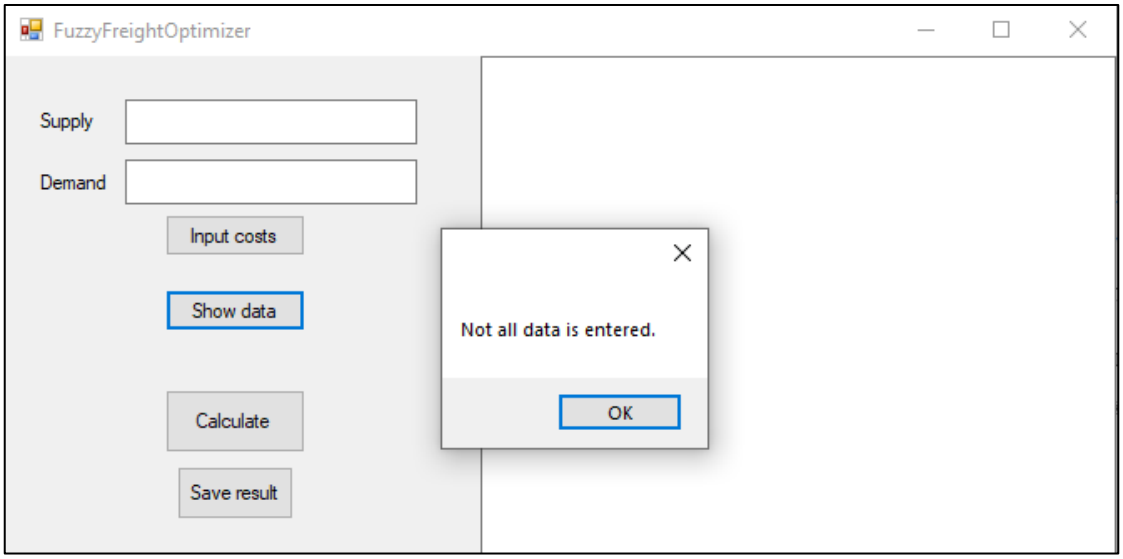

Рисунок 3.15 – Тестування кнопки Show data

На рисунку 3.15 можна побачити, що користувача було повідомлено про відсутність значень в полях supply та demand. Тому даний тест, як і інші, також можна вважати пройденим. При відсутності даних в тих самих полях, під час натискання кнопки Calculate буде отримано повідомлення (рис. 3.16).

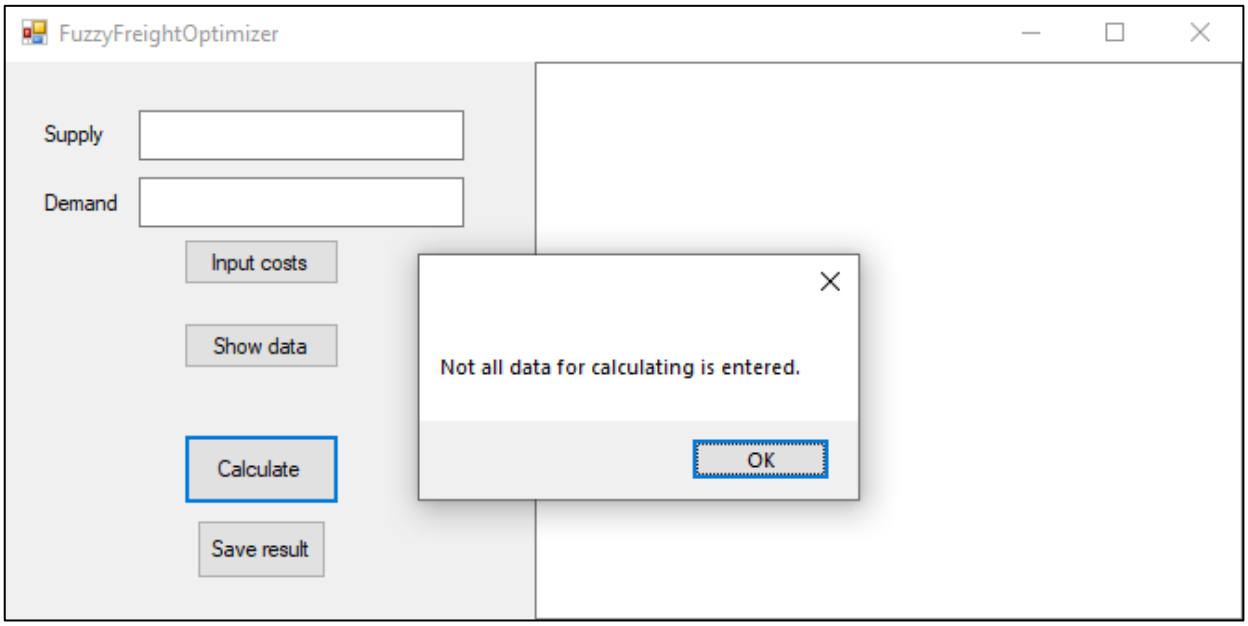

Рисунок 3.16 – Тестування кнопки Calculate

На останок залишилось перевірити роботу останньої кнопки – збереження результату. Для цього необхідно ввести всі дані стосовно перевезень та провести оптимізацію (рис. 3.11). Після чого, можна спробувати зберегти результат у форматі .txt.

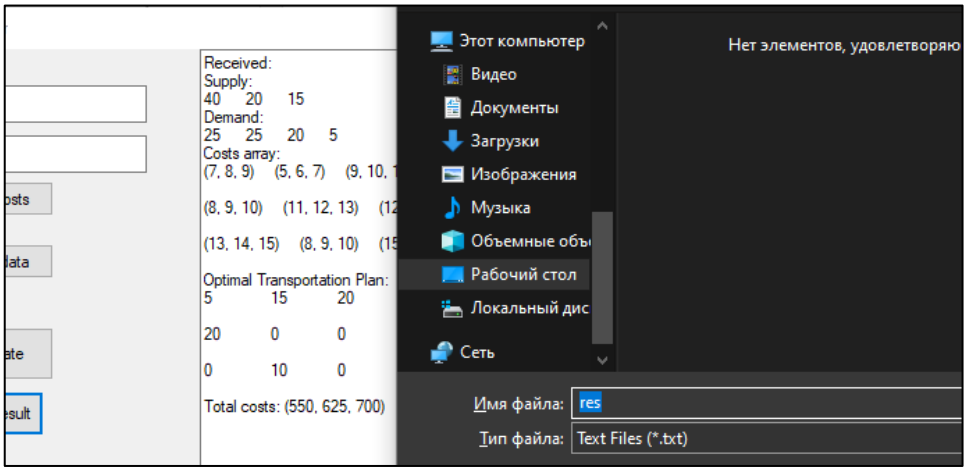

Рисунок 3.17 – Збереження результату в файл формату .txt

На рисунку 3.17 можна побачити, що кнопка збереження результату працює правильно. Вона збереже весь вміст того Textbox в файл, який створе користувач і повідомить його про успішність виконання даної операції.

В даному випадку, файл res.txt має такий вміст (рис. 3.18).

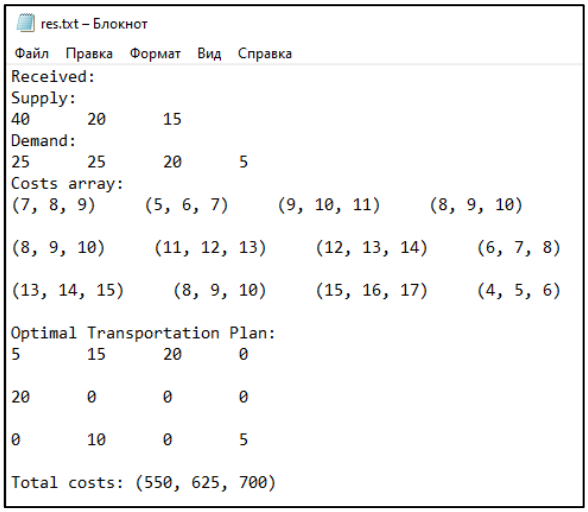

Рисунок 3.18 – Вміст файлу res.txt після збереження результатів

Маючи весь протестований функціонал, можна сказати, що даний застосунок працює так, як було заплановано при розробці.

Висновки до розділу 3

Під час написання третього розділу було обґрунтовано вибір інструмент розробки користувацьких інтерфейсів Windows Forms. Було описано, як з його використанням розроблено користувацький інтерфейс. Також, на мові програмування C#, було реалізовано можливість введення даних користувачем, метод Фогеля для побудови опорного плану та метод потенціалів для його оптимізації. В кінці розділу, було описано тестування реалізованого застосунка відповідно до вимог.

ВИСНОВКИ

В даній кваліфікаційній роботі бакалавра було реалізовано програмний застосунок задля вдосконалення процесу планування й оптимізації транспортних вантажоперевезень з метою підвищення їхньої ефективності та економічної доцільності.

Під час виконання роботи було досліджено сучасні фактори та проблеми, що прямо впливають на витрати транспортних вантажоперевезень; методи та підходи для планування і оптимізації вантажоперевезень, саме опираючись на них було обрано методи Фогеля та потенціалів.

Метою кваліфікаційної роботи було планування і оптимізація транспортних вантажоперевезень з нечіткими витратами шляхом створення відповідного програмного застосунку. Відповідно до мети, було виконано наступні поставлені завдання:

– проаналізовано становище та проблеми транспортних вантажоперевезень в сучасному світі;

– методи планування та оптимізації транспортних процесів;

– розроблено програмний застосунок для планування та оптимізації вантажоперевезень з урахуванням витрат;

– тестування розроблених інструментів на прикладі реальних виробничих ситуацій.

В результаті розроблено програмний застосунок для планування та оптимізації транспортних вантажоперевезень з нечіткими витратами, із застосуванням методів Фогеля та потенціалів. Користувач може ввести всі, необхідні для планування, дані та отримати оптимальний план перевезень, в якості результату, який можна зберегти в форматі .txt. Даний застосунок має інтуїтивно простий інтерфейс та функціонал, який чітко працює відповідно до технічного завдання.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Динаміка цін на пальне в Україні за останні роки. Аналіз ринку, прогнози. URL: https://ryderukraine.com/newsblog/dynamika-tsin-na-palne-v-ukraini-za-ostanni-roky-analiz-rynku-prohnozy/ (дата звернення: 07.05.2024).
2. What is GPS? URL: https://www.garmin.com/en-US/aboutgps/ (дата звернення: 07.05.2024).
3. Що таке інтернет речей, що про нього слідує знати. URL: https://habr.com/ru/companies/otus/articles/549550/ (дата звернення: 07.05.2024).
4. Хмарні технології: що це та які переваги надають людям. URL: https://gigacloud.ua/blog/navchannja/scho-take-hmarni-tehnologii (дата звернення: 07.05.2024).
5. Заборона дизельних авто в Європі. URL: https://ukr-prokat.com/blog/zaborona-dyzelnyh-avto-v-yevropi.html (дата звернення: 07.05.2024).
6. Під Києвом у величезному заторі застрягло близько тисячі фур. URL: https://www.unian.ua/economics/transport/zatori-u-stolici-pid-kiyevom-u-zatori-zastryaglo-blizko-tisyachi-fur-novini-kiyeva-11314799.html (дата звернення: 07.05.2024).
7. Носовська, О. Б., & Макаренко, М. В. (2014). Проблеми та перспективи розвитку транспортної інфраструктури України. Вісник Приазовського Державного Технічного Університету. Серія: Економічні науки, (27), 5–14 с. https://journals.uran.ua/ves_pstu/article/view/35575.
8. Реалізація транспортного потенціалу транспортної інфраструктури України в стратегії посткризового економічного розвитку. – К.: НІСД, 2011. – 37 с. (дата звернення: 07.05.2024).
9. Національна транспортна стратегія України на період до 2030 року. URL: https://publications.chamber.ua/2017/Infrastructure/UDD/National_Transport_Strategy_2030.pdf (дата звернення: 08.05.2024).
10. Сервіс моніторингу якості доріг. URL: https://www.autostrada.info/ua/map (дата звернення: 08.05.2024).

11. Мікуліна М. О. Проблеми та перспективи розвитку транспортної мережі України [Електронний ресурс] / М. О. Мікуліна, А. Д. Поливаний // Збірник тез доповідей по матеріалах 27-ї Міжнародної науково-практичної конференції «Технології XXI сторіччя», 24-26 лист. 2021 р. – Суми : СНАУ, 2021. – Ч. 1. – С. 152-154.

12. Fuzzy Logic Toolbox – MATLAB. URL: https://www.mathworks.com/products/fuzzy-logic.html (дата звернення: 12.05.2024).

13. Fuzzy Logic Toolbox documentation. URL: https://www.mathworks.com/help/fuzzy/ (дата звернення: 12.05.2024).

14. Надабан С., С. Дзітак, І. Дзітак. FuzzyTOPSIS: A General View. 2016 р. С. 823-831.

15. Махмуд М. Саліх, Б. Б. Зайдан, А.А. Зайдан. Survey on fuzzy TOPSIS state-of-the-art between 2007 and 2017. *Computers & Operations Research.* 2019 р. С. 207-227.

16. А. Чарнс, У. В. Купер. The Stepping Stone Method of Explaining Linear Programming Calculations in Transportation Problems. Management Science 1. 1954-1955 р. С. 49-69.

17. Harvey H. Shore. The Transportation Problem and the Vogel Approximation Method. *Decision Sciences*. Volume 1, Issue 3-4. July 1970. P. 441-457.

18. Шатрова І. А. Алгоритм оптимального розв’язання ресурсної задачі з використанням методу потенціалів транспортної задачі лінійного програмування / І. А. Шатрова, О. О. Демидова, С. В. Матвієвський // Наука та освіта : зб. пр. XVI Міжнар. наук. конф., 4–11 січ. 2022 р., м. Хайдусобосло, Угорщина. – Хмельницький : ХНУ, 2021. – С. 161-164.

19. Windows Forms documentation. URL: https://learn.microsoft.com/ru-ru/dotnet/desktop/winforms/?view=netdesktop-8.0 (дата звернення: 23.05.2024).

20. Вікно «Панель елементів» – Visual Studio. URL: https://learn.microsoft.com/ru-ru/visualstudio/ide/reference/toolbox?view=vs-2022 (дата звернення: 23.05.2024).

21. Learn about Solution Explorer –Visual Studio. URL: https://learn.microsoft.com/en-us/visualstudio/ide/use-solution-explorer?view=vs-2022 (дата звернення: 23.05.2024).

22. .NET Framework documentation. URL: https://learn.microsoft.com/en-us/dotnet/framework/ (дата звернення: 07.05.2024).

23. Документація з .NET. URL: https://learn.microsoft.com/ru-ru/dotnet/desktop/winforms/?view=netdesktop-8.0 (дата звернення: 08.06.2024).

ДОДАТОК А Реалізація сутності нечітких витрат на мові C#

```
internal class FuzzyNumber
{
    public double Lower { get; set; }
    public double Middle { get; set; }
    public double Upper { get; set; }

    public FuzzyNumber(double lower, double middle, double upper)
    {
        Lower = lower;
        Middle = middle;
        Upper = upper;
    }

    public static FuzzyNumber operator +(FuzzyNumber a, FuzzyNumber b)
    {
        return new FuzzyNumber(a.Lower + b.Lower, a.Middle + b.Middle,
a.Upper + b.Upper);
    }

    public static FuzzyNumber operator -(FuzzyNumber a, FuzzyNumber b)
    {
        return new FuzzyNumber(a.Lower - b.Lower, a.Middle - b.Middle,
a.Upper - b.Upper);
    }
    public double Defuzzify()
    {
        return (Lower + Middle + Upper) / 3.0;
    }
}
```

ДОДАТОК Б Реалізація методу Фогеля на мові C#

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace FuzzyFreightOptimizer
{
    internal class VogelMethod
    {
        public static void VogelApproximation(FuzzyNumber[,] costs, int[]
supply, int[] demand, int[,] result)
        {
            int rows = costs.GetLength(0);
            int cols = costs.GetLength(1);
            bool[] rowDone = new bool[rows];
            bool[] colDone = new bool[cols];

            while (true)
            {
                (double, int, int)[] rowPenalties = new (double, int,
int)[rows];
                (double, int, int)[] colPenalties = new (double, int,
int)[cols];

                for (int i = 0; i < rows; i++)
                {
                    if (!rowDone[i])
                    {
                        FuzzyNumber firstMin = new
FuzzyNumber(double.MaxValue, double.MaxValue, double.MaxValue);
                        FuzzyNumber secondMin = new
FuzzyNumber(double.MaxValue, double.MaxValue, double.MaxValue);
                        int firstIndex = -1;

                        for (int j = 0; j < cols; j++)
                        {
                            if (!colDone[j])
                            {
                                double cost = costs[i, j].Defuzzify();
                                if (cost < firstMin.Defuzzify())
                                {
                                    secondMin = firstMin;
                                    firstMin = costs[i, j];
                                    firstIndex = j;
```

```
                            }
                            else if (cost < secondMin.Defuzzify())
                            {
                                secondMin = costs[i, j];
                            }
                        }
                    }

                    rowPenalties[i] = (secondMin.Defuzzify() -
firstMin.Defuzzify(), i, firstIndex);
                }
            }

            for (int j = 0; j < cols; j++)
            {
                if (!colDone[j])
                {
                    FuzzyNumber firstMin = new
FuzzyNumber(double.MaxValue, double.MaxValue, double.MaxValue);
                    FuzzyNumber secondMin = new
FuzzyNumber(double.MaxValue, double.MaxValue, double.MaxValue);
                    int firstIndex = -1;

                    for (int i = 0; i < rows; i++)
                    {
                        if (!rowDone[i])
                        {
                            double cost = costs[i, j].Defuzzify();
                            if (cost < firstMin.Defuzzify())
                            {
                                secondMin = firstMin;
                                firstMin = costs[i, j];
                                firstIndex = i;
                            }
                            else if (cost < secondMin.Defuzzify())
                            {
                                secondMin = costs[i, j];
                            }
                        }
                    }

                    colPenalties[j] = (secondMin.Defuzzify() -
firstMin.Defuzzify(), j, firstIndex);
                }
            }

            var maxRowPenalty = rowPenalties.Where(p =>
!rowDone[p.Item2]).OrderByDescending(p => p.Item1).FirstOrDefault();
```

```
                var maxColPenalty = colPenalties.Where(p =>
!colDone[p.Item2]).OrderByDescending(p => p.Item1).FirstOrDefault();

                if (maxRowPenalty.Item1 <= 0 && maxColPenalty.Item1 <= 0)
                {
                    break;
                }

                int minI, minJ;
                if (maxRowPenalty.Item1 >= maxColPenalty.Item1)
                {
                    minI = maxRowPenalty.Item2;
                    minJ = maxRowPenalty.Item3;
                }
                else
                {
                    minJ = maxColPenalty.Item2;
                    minI = maxColPenalty.Item3;
                }

                int min = Math.Min(supply[minI], demand[minJ]);
                result[minI, minJ] = min;
                supply[minI] -= min;
                demand[minJ] -= min;

                if (supply[minI] == 0)
                {
                    rowDone[minI] = true;
                }
                if (demand[minJ] == 0)
                {
                    colDone[minJ] = true;
                }
            }
        }
    }
}
```

ДОДАТОК В Реалізація методу потенціалів на мові C#

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace FuzzyFreightOptimizer
{
    internal static class PotentialsMethod
    {
        public static int[,] SolveTransportationProblem(FuzzyNumber[,] costs,
int[] supply, int[] demand)
        {
            int rows = costs.GetLength(0);
            int cols = costs.GetLength(1);

            int[,] result = new int[rows, cols];
            VogelMethod.VogelApproximation(costs, supply, demand, result);

            AddDummyCellsIfNeeded(result);

            double[] u = new double[rows];
            double[] v = new double[cols];
            bool[] uSet = new bool[rows];
            bool[] vSet = new bool[cols];

            uSet[0] = true;
            while (true)
            {
                bool updated = false;
                for (int x = 0; x < rows; x++)
                {
                    for (int y = 0; y < cols; y++)
                    {
                        if (result[x, y] > 0)
                        {
                            if (uSet[x] && !vSet[y])
                            {
                                v[y] = costs[x, y].Defuzzify() - u[x];
                                vSet[y] = true;
                                updated = true;
                            }
                            else if (!uSet[x] && vSet[y])
                            {
                                u[x] = costs[x, y].Defuzzify() - v[y];
```

```
                            uSet[x] = true;
                            updated = true;
                        }
                    }
                }
            }
            if (!updated)
                break;
        }

        while (true)
        {
            double minDelta = double.MaxValue;
            int minI = -1, minJ = -1;
            for (int x = 0; x < rows; x++)
            {
                for (int y = 0; y < cols; y++)
                {
                    if (result[x, y] == 0 && uSet[x] && vSet[y])
                    {
                        double delta = costs[x, y].Defuzzify() - (u[x] +
v[y]);

                        if (delta < minDelta)
                        {
                            minDelta = delta;
                            minI = x;
                            minJ = y;
                        }
                    }
                }
            }

            if (minDelta >= 0)
                break;

            AdjustSolution(result, minI, minJ);
        }

        return result;
    }
    static void AddDummyCellsIfNeeded(int[,] result)
    {
        int rows = result.GetLength(0);
        int cols = result.GetLength(1);

        int filledCells = 0;
        for (int i = 0; i < rows; i++)
        {
```

```
            for (int j = 0; j < cols; j++)
            {
                if (result[i, j] > 0)
                {
                    filledCells++;
                }
            }
        }

        int requiredCells = rows + cols - 1;
        if (filledCells < requiredCells)
        {
            for (int i = 0; i < rows; i++)
            {
                for (int j = 0; j < cols; j++)
                {
                    if (result[i, j] == 0)
                    {
                        result[i, j] = 1;
                        filledCells++;
                        if (filledCells == requiredCells)
                        {
                            return;
                        }
                    }
                }
            }
        }
    }
    static void AdjustSolution(int[,] result, int startI, int startJ)
    {
        int rows = result.GetLength(0);
        int cols = result.GetLength(1);

        var path = new List<(int, int)>();
        path.Add((startI, startJ));

        var visited = new bool[rows, cols];
        visited[startI, startJ] = true;

        if (FindLoop(result, path, visited, startI, startJ))
        {
            int minValue = int.MaxValue;
            for (int k = 1; k < path.Count; k += 2)
            {
                int i = path[k].Item1;
                int j = path[k].Item2;
                minValue = Math.Min(minValue, result[i, j]);
```

```
                }

                for (int k = 0; k < path.Count; k++)
                {
                    int i = path[k].Item1;
                    int j = path[k].Item2;
                    if (k % 2 == 0)
                        result[i, j] += minValue;
                    else
                        result[i, j] -= minValue;
                }
            }
        }

        static bool FindLoop(int[,] result, List<(int, int)> path, bool[,]
visited, int currentI, int currentJ)
        {
            int rows = result.GetLength(0);
            int cols = result.GetLength(1);

            for (int i = 0; i < rows; i++)
            {
                if (result[i, currentJ] > 0 && !visited[i, currentJ])
                {
                    path.Add((i, currentJ));
                    visited[i, currentJ] = true;
                    if (i == path[0].Item1)
                        return true;
                    if (FindLoop(result, path, visited, i, currentJ))
                        return true;
                    path.RemoveAt(path.Count - 1);
                    visited[i, currentJ] = false;
                }
            }

            for (int j = 0; j < cols; j++)
            {
                if (result[currentI, j] > 0 && !visited[currentI, j])
                {
                    path.Add((currentI, j));
                    visited[currentI, j] = true;
                    if (j == path[0].Item2)
                        return true;
                    if (FindLoop(result, path, visited, currentI, j))
                        return true;
                    path.RemoveAt(path.Count - 1);
                    visited[currentI, j] = false;
                }
```

```
            }

            return false;
        }
    }
}
```

ДОДАТОК Г Вміст Form1.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using static System.Windows.Forms.VisualStyles.VisualStyleElement;

namespace FuzzyFreightOptimizer
{
    public partial class Form1 : Form
    {
        private int[] supply;
        private int[] demand;
        private FuzzyNumber[,] fuzzyCost;
        private FuzzyNumber totalCost;

        public Form1()
        {
            InitializeComponent();
            this.StartPosition = FormStartPosition.CenterScreen;
        }

        private void button1_Click(object sender, EventArgs e)
        {
            if (supply is null || demand is null || fuzzyCost is null)
            {
                MessageBox.Show("Not all data for calculating is entered.");
                return;
            }

            int[,] result =
PotentialsMethod.SolveTransportationProblem(fuzzyCost, supply, demand);
            TBOutput.Text += "Optimal Transportation Plan:\r\n";
            for (int i = 0; i < result.GetLength(0); i++)
            {
                for (int j = 0; j < result.GetLength(1); j++)
                {
                    TBOutput.Text += result[i, j] + "\t";
                }
                TBOutput.Text += "\r\n\r\n";
```

```
            }

            totalCost = new FuzzyNumber(0, 0, 0);
            for (int i = 0; i < result.GetLength(0); i++)
            {
                for (int j = 0; j < result.GetLength(1); j++)
                {
                    totalCost.Lower += result[i, j] * fuzzyCost[i, j].Lower;
                    totalCost.Middle += result[i, j] * fuzzyCost[i,
j].Middle;
                    totalCost.Upper += result[i, j] * fuzzyCost[i, j].Upper;
                }
            }
            TBOutput.Text += $"Total costs: ({totalCost.Lower},
{totalCost.Middle}, {totalCost.Upper})";
        }

        private void button2_Click(object sender, EventArgs e)
        {
            if (string.IsNullOrEmpty(TBSupply.Text))
            {
                MessageBox.Show("Enter supply data.");
                return;
            }
            if (string.IsNullOrEmpty(TBDemand.Text))
            {
                MessageBox.Show("Enter demand data.");
                return;
            }

            supply = ParseStringToIntArray(TBSupply.Text);
            demand = ParseStringToIntArray(TBDemand.Text);

            if(supply.Any(n => n < 0))
            {
                MessageBox.Show("Supply cannot contain negative values.");
                return;
            }

            if (demand.Any(n => n < 0))
            {
                MessageBox.Show("Demand cannot contain negative values.");
                return;
            }

            if (supply.Sum() != demand.Sum())
            {
                supply = null;
```

```
                demand = null;

                MessageBox.Show("The amount of supply must be equal to
demand.");
                return;
            }

            fuzzyCost = new FuzzyNumber[supply.Length, demand.Length];

            MatrixForm matrixForm = new MatrixForm(supply.Length + 1,
demand.Length);
            matrixForm.DataSubmitted += MatrixForm_DataSubmitted;
            matrixForm.ShowDialog();
        }
        private void button2_Click_1(object sender, EventArgs e)
        {
            SaveFileDialog saveFileDialog = new SaveFileDialog();
            saveFileDialog.Filter = "Text Files (*.txt)|*.txt|All Files
(*.*)|*.*";
            saveFileDialog.DefaultExt = "txt";
            saveFileDialog.AddExtension = true;

            if (saveFileDialog.ShowDialog() == DialogResult.OK)
            {
                try
                {
                    File.WriteAllText(saveFileDialog.FileName,
TBOutput.Text);
                    MessageBox.Show("The file was successfully saved!",
"Success", MessageBoxButtons.OK, MessageBoxIcon.Information);
                }
                catch (Exception ex)
                {
                    MessageBox.Show("Error saving file: " + ex.Message,
"Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
                }
            }
        }
        private void MatrixForm_DataSubmitted(object sender, FuzzyNumber[,]
data)
        {
            fuzzyCost = data;
        }

        public int[] ParseStringToIntArray(string input)
        {
            string[] parts = input.Split(new string[] { ", " },
StringSplitOptions.RemoveEmptyEntries);
```

```
            int[] result = new int[parts.Length];
            for (int i = 0; i < parts.Length; i++)
            {
                result[i] = int.Parse(parts[i]);
            }

            return result;
        }

        private void BTTNShowData_Click(object sender, EventArgs e)
        {
            if (supply is null || demand is null || fuzzyCost is null)
            {
                MessageBox.Show("Not all data is entered.");
                return;
            }

            TBOutput.Text += "Received:\r\n";
            TBOutput.Text += "Supply:\r\n";
            for (int i = 0; i < supply.Length; i++)
            {
                TBOutput.Text += supply[i] + "      ";
            }
            TBOutput.Text += "\r\n";

            TBOutput.Text += "Demand:\r\n";
            for (int i = 0; i < demand.Length; i++)
            {
                TBOutput.Text += demand[i] + "      ";
            }
            TBOutput.Text += "\r\n";

            TBOutput.Text += "Costs array:\r\n";
            for (int i = 0; i < fuzzyCost.GetLength(0); i++)
            {
                for (int j = 0; j < fuzzyCost.GetLength(1); j++)
                {
                    TBOutput.Text += $"({fuzzyCost[i, j].Lower},
{fuzzyCost[i, j].Middle}, {fuzzyCost[i, j].Upper})" + "      ";
                }
                TBOutput.Text += "\r\n\r\n";
            }
        }
    }
}
```

ДОДАТОК Д Вміст MatrixForm.cs

```
using System;
using System.Collections.Generic;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Text.RegularExpressions;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace FuzzyFreightOptimizer
{
    internal class MatrixForm : Form
    {
        private DataGridView dataGridView;
        private Button btnSubmit;
        private TableLayoutPanel tableLayoutPanel;

        public event EventHandler<FuzzyNumber[,]> DataSubmitted;

        public MatrixForm(int rows, int columns)
        {
            InitializeComponent(rows, columns);
        }

        private void InitializeComponent(int rows, int columns)
        {
            this.dataGridView = new DataGridView();
            this.btnSubmit = new Button();
            this.tableLayoutPanel = new TableLayoutPanel();

            //
            // tableLayoutPanel
            //
            this.tableLayoutPanel.ColumnCount = 1;
            this.tableLayoutPanel.RowCount = 2;
            this.tableLayoutPanel.Dock = DockStyle.Fill;
            this.tableLayoutPanel.RowStyles.Add(new
RowStyle(SizeType.Percent, 90F));
            this.tableLayoutPanel.RowStyles.Add(new
RowStyle(SizeType.Percent, 10F));

            //
            // dataGridView
            //
```

```
            this.dataGridView.ColumnHeadersHeightSizeMode =
DataGridViewColumnHeadersHeightSizeMode.AutoSize;
            this.dataGridView.ColumnCount = columns;
            this.dataGridView.RowCount = rows;
            this.dataGridView.Dock = DockStyle.Fill;

            this.dataGridView.AllowUserToAddRows = false;

            this.dataGridView.RowTemplate.Height = 30;
            this.dataGridView.ColumnHeadersHeight = 30;

            this.dataGridView.DefaultCellStyle.Font = new Font("Arial", 12);
            this.dataGridView.ColumnHeadersDefaultCellStyle.Font = new
Font("Arial", 12);

            //
            // btnSubmit
            //
            this.btnSubmit.Text = "Submit";
            this.btnSubmit.Dock = DockStyle.Fill;
            this.btnSubmit.Click += new EventHandler(this.BtnSubmit_Click);

            //
            // MatrixForm
            //
            this.ClientSize = new System.Drawing.Size(620, 460);
            this.Controls.Add(this.tableLayoutPanel);
            this.tableLayoutPanel.Controls.Add(this.dataGridView, 0, 0);
            this.tableLayoutPanel.Controls.Add(this.btnSubmit, 0, 1);
            this.StartPosition = FormStartPosition.CenterScreen;
            this.Text = "Matrix Costs Input Form";
        }
        private void BtnSubmit_Click(object sender, EventArgs e)
        {
            int rows = dataGridView.RowCount;
            int columns = dataGridView.ColumnCount;
            FuzzyNumber[,] data = new FuzzyNumber[rows, columns];

            for (int i = 0; i < rows; i++)
            {
                for (int j = 0; j < columns; j++)
                {
                    if (!ValidFormatChecker.IsValidFormat(dataGridView[j,
i].Value.ToString()))
                    {
                        MessageBox.Show("Some data entered incorrectly.");
                        return;
                    }
```

```
                    string[] values = dataGridView[j,
i].Value.ToString().Trim('(', ')').Split(',');
                    double low = Convert.ToDouble(values[0]);
                    double mid = Convert.ToDouble(values[1]);
                    double high = Convert.ToDouble(values[2]);
                    data[i, j] = new FuzzyNumber(low, mid, high);
                }
            }

            DataSubmitted?.Invoke(this, data);
            this.Close();
        }

    }
}
```