МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

КАФЕДРА СИСТЕМНОГО АНАЛІЗУ ТА ОБЧИСЛЮВАЛЬНОЇ МАТЕМАТИКИ
(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)
магістр
(ступінь вищої освіти)

на тему Дослідження впливу невизначеності на оптимальний розв'язок задачі про рюкзак

Виконав: студент(ка) 2 курсу, групи КНТ-811м

Спеціальності 124 – Системний аналіз
(код і найменування спеціальності)

Освітня програма (спеціалізація)
«Інтелектуальні технології та прийняття рішень в складних системах»

Смола В.І.
(прізвище та ініціали)
Керівник Подковаліхіна О.О.
(прізвище та ініціали)
Рецензент Лозовська Л.І.
(прізвище та ініціали)

2022

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
(повне найменування закладу вищої освіти)

Кафедра Системного аналізу та обчислювальної математики
Ступінь вищої освіти магістр
Спеціальність 124 – Системний аналіз
(код і найменування)
Освітня програма
(спеціалізація) «Інтелектуальні технології та прийняття рішень в складних системах»
(назва освітньої програми (спеціалізації)

ЗАТВЕРДЖУЮ
Завідувач кафедри _____
_____Г.В.Корніч_____
«_____» _____20___року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

Смоли Владислава Ігоровича
(прізвище, ім'я, по батькові)

1. Тема проєкту (роботи) Дослідження впливу невизначеності на оптимальний розв'язок задачі про рюкзак
керівник проєкту (роботи) Подковаліхіна Олена Олександрівна, к.ф.-м.н., доц. ,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом закладу вищої освіти від « 28 » листопада 2022 року № 407
2. Строк подання студентом проєкту (роботи) « 19 » грудня 2022 року
3. Вихідні дані до проєкту (роботи) Умови задачі пакування рюкзаку, перелік розглянутих робіт про задачу пакування рюкзака.
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) В першому розділі проаналізовані роботи авторів, що розглядали задачу про рюкзак. В другому розділі наведено загальну та математичну постановки задачі упакування рюкзака з вхідними параметрами, що утворюють систему незалежних випадкових величин. Описано підхід для знаходження оптимальних розв'язків задачі. Розроблено застосунок на мові програмування C++, що має інтерфейс та знаходить оптимальні розв'язки вище вказаної задачі та відповідну їм сумарну вартість, та створює файл, в який записуються отримані дані для заданої кількості експериментів. Розроблені універсальні критерії оптимальності. В третьому розділі продемонстровані приклади задачі про рюкзак. Отримані та проаналізовані оптимальні розв'язки першої задачі з варіюванням вартості згідно нормальному та рівномірному законам розподілу. Отримані та проаналізовані оптимальні розв'язки другої задачі з варіюванням об'єму згідно нормальному та рівномірному законам розподілу. Отримані та проаналізовані оптимальні розв'язки третьої задачі з варіюванням вартості та об'єму згідно нормальному та рівномірному законам розподілу.

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

_____

6. Консультанти розділів проєкту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1	Подковаліхіна О.О., к.ф.-м.н., доц.		
2	Подковаліхіна О.О., к.ф.-м.н., доц.		
3	Подковаліхіна О.О., к.ф.-м.н., доц.		
Нормоконтроль	Широкорад Д.В., к.ф.-м.н., ст. викл.		

7. Дата видачі завдання «02» вересня 2022 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Сформулювати мету на основні завдання дипломної роботи	02.09.2022 – 09.09.2022	
2	Опрацювати роботи авторів за темою дослідження	12.09.2022 – 30.09.2022	
3	Розробка застосунку для розв'язку задачі	03.10.2022 – 31.10.2022	
4	Адаптування застосунку	01.11.2022 – 11.11.2022	
5	Розрахунки та аналіз даних	12.11.2022 – 17.11.2022	
6	Оформлення пояснювальної записки	18.11.2022 – 13.12.2022	
7	Попередній захист дипломної роботи та отримання рецензій	13.12.2022 – 18.12.2022	
8	Захист дипломної роботи	19.12.2022	

Студент(ка) _____ Смола В.І.
(підпис) (прізвище та ініціали)

Керівник проєкту (роботи) _____ Подковаліхіна О.О.
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Дипломна робота: 62 с., 2 рис., 15 табл., 1 дод., 13 літературних джерел.

Об’єкт дослідження – задача пакування рюкзака.

Предмет дослідження – вплив статистичної невизначеності на оптимальний розв’язок задачі про рюкзак.

Мета роботи – розроблення підходу знаходження оптимального розв’язку задачі, вхідні параметри якої є системою незалежних випадкових величин і розроблення застосунку з інтерфейсом, який знаходить оптимальні розв’язки та сумарну вартість і записує результати в файл з заданою кількістю експериментів та дослідження впливу статистичної невизначеності за рівномірним і нормальним законами розподілу на оптимальний розв’язок задачі про рюкзак.

Методи дослідження – метод повного перебору.

В дипломній роботі проаналізовані роботи авторів, що розглядали задачу про рюкзак. Наведено загальну та математичну постановки задачі упакування рюкзака з вхідними параметрами, що утворюють систему незалежних випадкових величин. Описано підхід для знаходження оптимальних розв’язків задачі. Розроблено застосунок на мові програмування C++, що має інтерфейс та знаходить оптимальні розв’язки вище вказаної задачі та відповідну їм сумарну вартість, та створює файл, в який записуються отримані дані для заданої кількості експериментів. Розроблені універсальні критерії оптимальності. Продемонстровані приклади задачі про рюкзак. Отримані та проаналізовані оптимальні розв’язки першої задачі з варіюванням вартості, другої задачі з варіюванням об’єму і третьої задачі з варіюванням вартості та об’єму згідно нормальному та рівномірному законам розподілу.

УПАКУВАННЯ РЮКЗАКА, УМОВА НЕВИЗНАЧЕНОСТІ, ЗАДАЧА ПРО РЮКЗАК, МЕТОД ПОВНОГО ПЕРЕБОРУ, КРИТЕРІЇ ОПТИМАЛЬНОСТІ, МОВА ПРОГРАМУВАННЯ C++.

ЗМІСТ

ВСТУП

У повсякденному житті постійно знаходимо рішення задач і завжди намагаємося зробити ці рішення оптимальними. У разі простих задач вирішення зазвичай приймаються негайно, а у разі більш складних задач вирішення іноді потребує багато часу. Часто при прийнятті таких рішень ми моделюємо завдання з використанням математичних понять, а потім визначаємо рішення отриманої моделі за допомогою комп'ютера, використовуючи правильно вибрані методи.

Однією з проблем, яку ми можемо вирішити описаним способом, є проблема рюкзака, яку ми розглянемо в цій роботі. Проблема оптимізації, яка називається проблемою «рюкзака», взяла свою назву з аналогії з практичною ситуацією, подібною до проблеми упаковки рюкзака. Йдеться про те, щоб упакувати якнайбільше цінних предметів та не перевищити вантажопідйомності рюкзаку. Задача про рюкзак – практична задача, з якою стикаються навіть транспортні компанії. У цьому випадку все залежить від оптимального завантаження не тільки предметів певної ваги в транспортний засіб, але також від належного розподілу цих предметів, щоб вони могли уміститися якомога більше.

Задача пакування рюкзака застосовувалася в логістиці, криптографії, економіці та прикладній математиці, але зараз почали використовувати на підприємствах машинобудівництва, для розподілу ресурсів сервера, для формування команди стартап-проекту та навіть для визначення набору вітрових електроустановок ВЕС. Таким чином розширення застосування в різних сферах цієї задачі показує її незгасаючу актуальність.

Вміння знаходити оптимальні розв’язки, коли зміна підпорядковується закону розподілу, грає важливу роль для побудування порівняльної оцінки і тому було вирішено писати на дану тему дипломну роботу.

1 ОГЛЯД ЛІТЕРАТУРИ ТА ОСТАННІХ ДОСЛІДЖЕНЬ І ПУБЛІКАЦІЙ

Задача про рюкзак є, майже, в кожній сфері діяльності людини. Але частіш за все дана задача використовується в логістиці. В наші часи логістика відіграє важливу роль, як в грошовому плані, так і в швидкодії роботи транспорту. Завдяки задачі про рюкзак, або задачі про ранець, можна скоротити час на знаходження різних ефективних рішень для вирішення логістичних проблем.

Тому різними логістичними компаніями були створені спеціальні програми для оптимізації вантажу, маршруту тощо.

Дану класичну задачу розглядали декілька століть різні автори і писали в роботах, якими саме методами вирішували проблематику даної задачі опираючись на швидкодію даних методів та їх зручного використання.

Задача про рюкзак має різні формулювання такі як:

- Задача розподілу ресурсів (задача розподілення інвестицій).
- Задача упакування рюкзаку(контейнера, корабля, літака тощо).

Раніше цю задачу застосовували в таких сферах, які будуть описані в роботах нижче.

В статті [1] Березового М.І., Пожидаєва С.О., Бурова В.С. описано використання методу гілок та меж для визначення оптимального варіанту розподілу сортувальних колій між призначеннями вагонопотоків. Ця задача є задачею розподілу ресурсів. Частіш за все, така задача є цілочисельною задачею лінійного програмування. В статті проаналізовано, що метод гілок і меж є ефективним в цьому випадку та описується алгоритм послідовних дій для даної задачі.

В роботі [2] Концеби С.М. описана економіко-математична модель оптимізаційної задачі плодоовочесховища у сільськогосподарській сфері, яка є

поетапною задачею стохастичного програмування з обмеженнями. Дана модель дає можливість оптимізувати структуру плодоовочевої продукції на збереженні за її видами та розподіл капіталовкладень за основними напрямами.

В статті [3] Медиковського М.О., Теслюка В.М. та Шуневича О.Б. описано підхід для знаходження плану вітрових електроустановок ВЕС (вітрова електростанція), який базується на використанні методу динамічного програмування і розроблених програмних засобів. На першому етапі задаються початкові умови системи, що передбачає визначення початкового стану кожної з ВЕУ (вітрова електроустановка), а саме: задання початкових параметрів для кількості виробленої енергії, часу напрацювання кожної ВЕУ, кількості комутації та інших параметрів, що визначають технічний стан кожної ВЕУ. Потім можна розв'язати задачу про рюкзак методом динамічного програмування, який передбачає:

$$b_1 P_1 + b_2 P_2 + \cdots + b_m P_m \leq P,$$
$$b_1 C_1 + b_2 C_2 + \cdots + b_m C_m \to \max \quad ,$$

де $b_1, b_2, \ldots, b_m$ – набір бінарних величин, які необхідно визначити;

$P_j > 0$ – потужність кожної ВЕУ;

$C_j > 0$ – ефективність (інтегральна оцінка) кожної ВЕУ;

P – задана потужність ВЕС, яка необхідна в даний час.

Для проведення експерименту кількість ітерація була обрана 8640 – це показник роботи ВЕС за один рік. Потужності обиралися випадковим чином в діапазоні від 100 до 2500 кВт. Сумарна потужність ВЕС дорівнює 70000 кВт. Також в статті були обрані критерії ефективності: час напрацювання турбін, вироблена потужність та число комутацій.

В результаті досліджень розроблено метод визначення динамічної структури вітрової електростанції з використанням узагальнюючого критерію ефективності ВЕУ та засобів розв'язання оптимізаційних задач.

В роботі [4] Маслової Н.А., Мовчана О.В. розглядається застосунок, в якому є заготовлений код програми на мові програмування C# для розв’язання різних задач математичного програмування. В цій програмі розглянутий певний спектр задач, а саме задача розподілу ресурсів. В розробленому авторами застосунку є різний функціонал такий, як рішення класичних задач розподілу ресурсів, мережеві задачі розподілу ресурсів та різні методи і алгоритми їх вирішення. Структура меню застосунку виглядає таким чином:

- рішення класичних задач:
  - задача розподілу коштів;
  - задача управління запасами;
- мережева задача розподілу ресурсів:
  - задача про найкоротший шлях;
  - задача про максимальний потік;
- методи і алгоритми:
  - метод прямої прогонки;
  - метод зворотної прогонки;
  - метод гілок і меж;
  - метод проєкції градієнта;
  - алгоритм Балаша;
  - алгоритм Качмажа.

Потім можна побачити в застосунку базову постановку задачі для кожної теми, а також приклад вирішення, програмний код та опис вирішення.

Задача розподілу ресурсів є одною із головних розділів в розв’язку багатьох наукових задач сучасності. І тому було вирішено розробити даний застосунок, щоб користувач міг легко отримати розв’язок необхідної задачі з описом цього вирішення і рекомендаціями для вибору найбільш відповідного методу вирішення.

В роботі [5] Яковлевої А.П., Курдупа І.О. описано задачу для знаходження оптимального плану розподілу ресурсів між підприємствами та

розглянуті різні варіації формулювання даної задачі. Методом динамічного програмування знаходять розв'язки. Математично за принципом оптимальності Беллмана оптимальна стратегія описується таким чином:

$$B_{N-1}(S_l) = optimum\left[R_{l+1}(S_l, U_{l+1}) + B_{N-(l+1)}(S_{l+1})\right], l = \overline{0, N-1},$$

де U_l – керування, обране на l-му кроці; стан системи на l-му кроці; R_l – безпосередній ефект, досягнутий на l-му кроці;B_{N-1} – оптимальне значення ефекту, що досягається за $N - l$ кроків;N – загальна кількість кроків.

Тому задача про рюкзак не втрачає актуальності навіть в наші часи. І навіть з'явились нові сфери, де дана задача використовується і буде описана в роботах нижче.

В роботі [6] автор Кравченко Є.І. розглядає задачу в такій інтерпретації: «Припустимо, є N кандидатів для участі в стартап-проєкті, де $i = \overline{1, N}$ індекс кандидатів, які є учасниками проєкту, x_i – кількість проєктів, в яких брали участь кожен i-ий учасник відповідно до зазначених в даній вакансії компетенціях; w_i – заробітна плата, яку бажає отримати кандидат, W – максимальний капітал заробітної плати.

Ціль визначити учасників проєкту, що мають найбільший досвід роботи, але вимагають невелику заробітну плату.» .

Взагалі задачу пакування рюкзака можна поділити на декілька різних видів в залежності від ситуації. Основними видами задачі є:

- 0-1 задача пакування рюкзака.
- Обмежена задача пакування рюкзака.
- Необмежена задача пакування рюкзака.
- Рюкзак з мультивибором.
- Мультиплікативний рюкзак.
- Багатовимірний рюкзак.

Це самі поширеніші види задачі про рюкзак.

В роботі [7] Ткачука В.М. розглядається задача про мультиплікативний рюкзак. І суть цієї задачі полягає у розбитті множини n предметів на m підмножини, які не перетинаються, і в результаті отримуємо максимальну сумарну вартість упакованих предметів. І також об'єм кожного із наборів не може перевищувати максимальну ємність відповідного ранця P_i. Формально задача зводиться до знаходження такої матриці заповнення $X: x_{i,j} \in \{1, \dots, m\}, j \in \{1, \dots, m\}$, яка забезпечує максимальне значення функції:

$$F(x) = \sum_{i=1}^{m} \sum_{j=1}^{n} \omega_i x_{i,j}$$

та задовільняє обмеження:

$$\sum_{i=1}^{m} p_i x_{i,j} \leq P_j \, , j \in \{1, \dots, m\},$$

де $\omega_i -$ вартість i-того предмету;

p_i – вага i-того предмету.

Автор в роботі пише, що в одному випадку ефективним методом вирішення даної задачі є еврестичні алгоритми, які дозволяють отримати близькі оптимальні плани для задачі з великими обсягом за прийнятний час. І одним із кращих методів знаходження розв'язку даної задачі є квантовий генетичний алгоритм (QGA), робота якого ґрунтується на поєднанні ідей квантових обчислень та технології класичних генетичних алгоритмів.

В роботі [8] Зленко Ю. розповідається про актуальність задач динамічного програмування та про їх математичну складову. Еволюція сучасного суспільства досягла такого етапу, коли виникає гостра необхідність у розробці ефективних способів організації систем різного призначення та рівня для знаходження найефективнішого плану досягнення бажаних результатів. Виконання емпіричних досліджень відповідних процесів та явищ є

недоцільними з багатьох причин: вартість дослідження, їх динамічність та унікальність тощо. Тому ці дослідження замінюються аналізом відповідних математичних моделей.

В різних сферах виникають схожі задачі, які є подібними за постановкою та мають спільні ознаки. І досить поширеною є задача про рюкзак, яка формулюється дуже коротке і просто: рюкзак (вагон, судно, літак) має обмежену вантажність. Потрібно заповнити рюкзак так, щоб отримати максимальний прибуток.

В статті [9] розглядається наукова новинка автором Качор П.М. , а саме розглядається задача про рюкзак для знаходження розв'язку проблеми оптимального плану для розподілення присутніх ресурсів веб-сервера, враховуючи неможливе їх збільшення. Дану проблематику в загальному вигляді формулюють так: є N предметів, n_i предмет має масу $w_i > 0$ і вартість $p_i > 0$. Потрібно обрати з цих предметів такий план, який не перевищує задану величину W (місткість ранця), а сумарна вартість має бути максимальною. Є деякі методи для знаходження розв'язку, і одним із кращих методів є метод динамічного програмування.

Роботу методу динамічного програмування автор описує так: є $A(i, s)$, що дорівнює максимальній вартості речей, які є в сумці об'ємом s, при умові використання перших i предметів. При цьому

$$A(i, 0) = 0$$
$$A(0, s) = 0$$

мається на увазі, що при відсутності речей, або їх нульвої вартості, їх сумарна вартість буде дорівнювати 0.

Задача вирішується методом зведення до простої задачі. На кожному кроці вибираємо чи є i-тий предмет у сумці(ранці, рюкзаку). Крім того, якщо i-ті предмети знаходяться в сумці, то $A(i, s)$ проблема зводиться до простого

рішення, де місткість зменшується на об'єм w_k i-того предмету, а кількість речей зменшується на один. Тим часом вартість сумки збільшується на вартість p_k даного предмету. Якщо i-тий елемент не є в сумці, то його вартість не змінюється, а вибір продовжується з $i-1$ елемента.

Тому розв'язок даної задачі зводиться до рекурентної формули:

$$A(i,s) = \max(A(k-1,s), A(k-1,s-w_k) + p_k)$$

Інакше кажучи, визначаємо, який оптимальний план є найціннішим – той, що містить даний предмет, чи без нього.

Пошук предметів або елементів, що кладуться до сумки, закінчується на етапі заповнення сумки повністю.

В роботі [10] Рибачкової І.А. розглядається задача про рюкзак, але формулювання задачі звучить так: упакування куль у опуклому багатограннику із зонами заборони. Таку тему було обрано з метою побудови ефективного методу розв'язання задачі упаковки куль та дослідження етапів переходу від однієї точки локального максимуму до іншого, таким чином, щоб наповненість отримала максимальне значення.

В роботі [11] Безсонного В.Л., Пономаренка Р.В., Третьякова О.В., Карпеця К.М. розробляють і описують алгоритм оптимального управління ризиками небезпечних подій на машинобудівному підприємстві. І розглядаються 3 стратегії ризику: прийняття ризику, уникнення ризику, управління ризиком. Необхідно не ухилятись від ризику, а вміти ним користуватись та оцінювати і не виходити за межі.

Для того, щоб знайти значення ризику знайдемо матрицю, рядки якої дорівнюють різним ймовірностям, а стовпчики – різним збиткам. Кожна клітинка матриці дорівнює значенню ризику. Поділимо на три рівні ймовірності та збитки: високий, середній та мінімальний.

Наприклад, є задача управління ризиком – зменшити ризик з мінімальнішими витратами. Зменшення ризику роблять завдяки двом процесам:

- Перший процес: зменшення ймовірності настання небажаної події.
- Другий процес: зменшення збитку при настанні небажаної події.

Візьмемо спочатку, що ці процеси не пересікаються.

Припустимо, є n процесів першого типу. Позначимо a_i – зменшення ймовірності p при проведенні i-того процесу, b_i – витрати на проведення i-того процесу. Тепер позначимо A_1 – значення для зменшення ймовірності необхідної для переводу даного показника в категорію мінімуму, A_2 – значення для зменшення ймовірності необхідної для переводу даного показника в категорію середнього ризику. Позначимо $x_i = 1$, якщо i-й процес увійшов в програму зниження ризику, $x_i = 0$ в іншому випадку.

Постановка завдання:

Визначити $x_i, i = \overline{1,n}$, такі що $\sum_i \ b_i x_i \to \min$,

при обмеженнях $\sum_i \ a_i x_i \geq A_1$

Це задача про рюкзак, що ефективно вирішується методом дихотомічного програмування при цілочислових значеннях параметрів.

2 ЗАДАЧА ПРО РЮКЗАК В УМОВАХ СТАТИСТИЧНОЇ НЕВИЗНАЧЕНОСТ

2.1 Загальна постановка задачі про рюкзак

Розглянемо задачу оптимізації упакування рюкзака з вхідними параметрами, що утворюють систему незалежних випадкових величин. В тривимірний простір (рюкзак, мішок, кузов автомобіля тощо) певного об'єму (ваги), який не має перевищувати P умовних одиниць (у.о.), упаковуються предмети n типів. Кожний предмет має свій об'єм (вагу) v_i і вартість (цінність) s_i. Вартість (цінність) або/та об'єм (вага) одного предмету кожного типу може варіюватися за відповідним законом розподілу. Потрібно упакувати предмети в тривимірний простір таким чином, щоб сумарна вартість (цінність) цих предметів була максимальною і не перевищувала об'єм (вагу) простору.

2.2 Математична постановка задачі про рюкзак

$$F = \sum_{i=1}^{n} s_i x_i \to max$$

$$\sum_{i=1}^{n} v_i x_i \leq P,$$

v_i можуть варіюватися відповідно заданому закону розподілу або залишатися сталими,

s_i можуть варіюватися відповідно заданому закону розподілу або залишатися сталими,

де n – загальна кількість предметів;

x_i – кількість предметів типу i;

s_i – вартість (цінність) одного упакованого предмету типу i;

v_i – вага (об’єм) одного упакованого предмету типу i;

P – вага (об’єм) рюкзаку;

F – сумарна вартість (цінність) упакованих предметів без перевищення об’єму (ваги) рюкзаку.

2.3 Підхід знаходження розв’язків задачі

1. Оберемо тип задачі та певний закон розподілу.

Першою задачею будемо називати задачу про рюкзак, в якій вартість може варіюватися відповідно заданому закону розподілу, а об’єм залишається сталим. Другою задачею будемо називати задачу про рюкзак, в якій об’єм може варіюватися відповідно заданому закону розподілу, а вартість залишається сталим. Третьою задачею називаємо задачу про рюкзак, в якій об’єм та вартість може варіюватися відповідно заданому закону розподілу. Таким чином були описані всі «типи задач» в алгоритмі.

За алгоритмом можна обрати нормальний або рівномірний закон розподілу.

2. Введемо або оберемо такі вхідні дані:

- Об’єм (вага) рюкзаку.
- Об’єм (вага) предмету кожного типу.
- Вартість (цінність) предмету кожного типу.
- Кількість експериментів.
- Відхилення вартості або/та об’єму предмету кожного типу.
- Шлях знаходження файлу.

3. Прораховуємо максимальну кількість упакованих предметів, що можуть поміститься в рюкзак для предметів кожного типу.

4. Необхідно знайти кількість всіх можливих планів, що можуть поміститься в рюкзак враховуючи всі умови задачі.

Кількість всіх можливих планів рахується для того, щоб в застосунку на майбутнє виділить місце або створити масив, де будуть зберігатися всі оптимальні плани. Додатково виділяється місце або створюється масив, де будуть зберігатися результати, а саме сумарна вартість під кожним планом, що дає для поточної ітерації максимальну сумарну вартість.

5. Запишемо всі можливі плани в виділене місце або в створений масив, що можуть поміститься в рюкзак враховуючи всі умови задачі.

6. Генеруємо випадкові числа вартості (цінності) та/або об’єму (ваги) за обраним законом розподілом враховуючи відсоток відхилення для кожного типу предметів.

Завдяки вбудованим функціям генеруємо значення за певним законом розподілу вводячи в неї вхідні параметри для генерації.

7. Використовуємо метод повного перебору для знаходження найкращого оптимального плану, який дає максимальну сумарну вартість для даних випадкових величин.

В даному кроці робиться повний перебір всіх планів так, що можна точно знайти оптимальний план, що дає максимальну сумарну вартість в поточну ітерацію.

8. Записуємо знайдену сумарну вартість під певний оптимальний план в виділене місце або в створений масив.

9. Повторюємо кроки 6-8 до тих пір, поки не виконаємо обрану кількість експериментів.

10. Записуємо всі оптимальні плани, що з’явились, в файл.

11. Записуємо всі отримані сумарні вартості кожного оптимального плану в файл.

2.4 Програмна реалізація для знаходження розв’язку задачі

Програмний код було розроблено на мові програмування C++ [12]. Для розробки застосунку було використано код програми з минулої роботи цього ж автора [13]. В додатку наведено код авторської програми (Додаток А).

За завданням потрібно було розробити застосунок, який би знаходив всі оптимальні плани та їх сумарну вартість для заданої кількості експериментів для певної задачі про рюкзак. Але для того, щоб користувачеві було краще зрозуміло, як і що, де змінювати, а також, щоб було зручно використовувати застосунок, для даної програми було вирішено додати інтерфейс.

Для розробки інтерфейсу програми була обрана платформа WinForms, яка взаємодіє з мовою програмування C++. Завдяки цій платформі організовуємо інтерфейс користувача.

В сам інтерфейс було додано такі моменти:

- Вибір задачі (різні варіації задачі).
- Вибір розподілу (нормальний, рівномірний).
- Введення початкових умов в категорії вартості.
- Введення початкових умов в категорії об’єму.
- Введення максимального об’єму рюкзаку.
- Введення у відсотках відхилень вартості для кожного предмету.
- Введення у відсотках відхилень об’єму для кожного предмету.
- Введення кількості експериментів.
- Обирання шляху для збереження файлу.
- Кнопка «Запуск»
- Шкала, яка показує завершення операції.

Далі буде продемонстровано інтерфейс застосунку (рис 2.1) та інструкція по використанню.

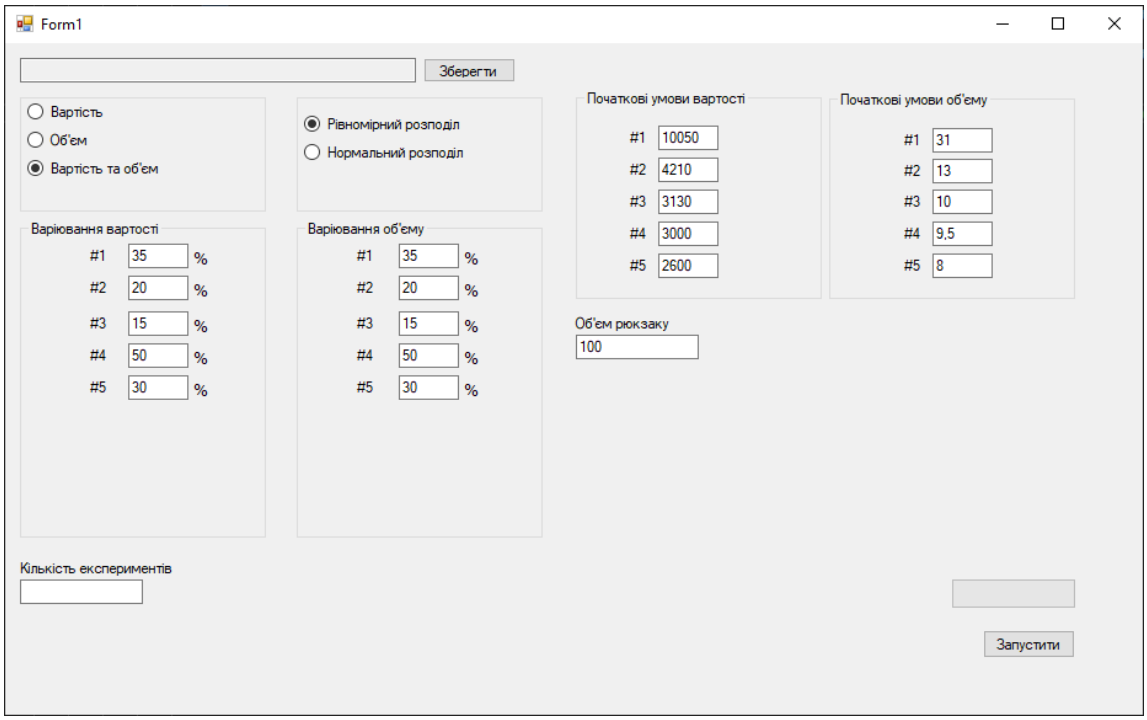

Рисунок 2.1 – Інтерфейс застосунку

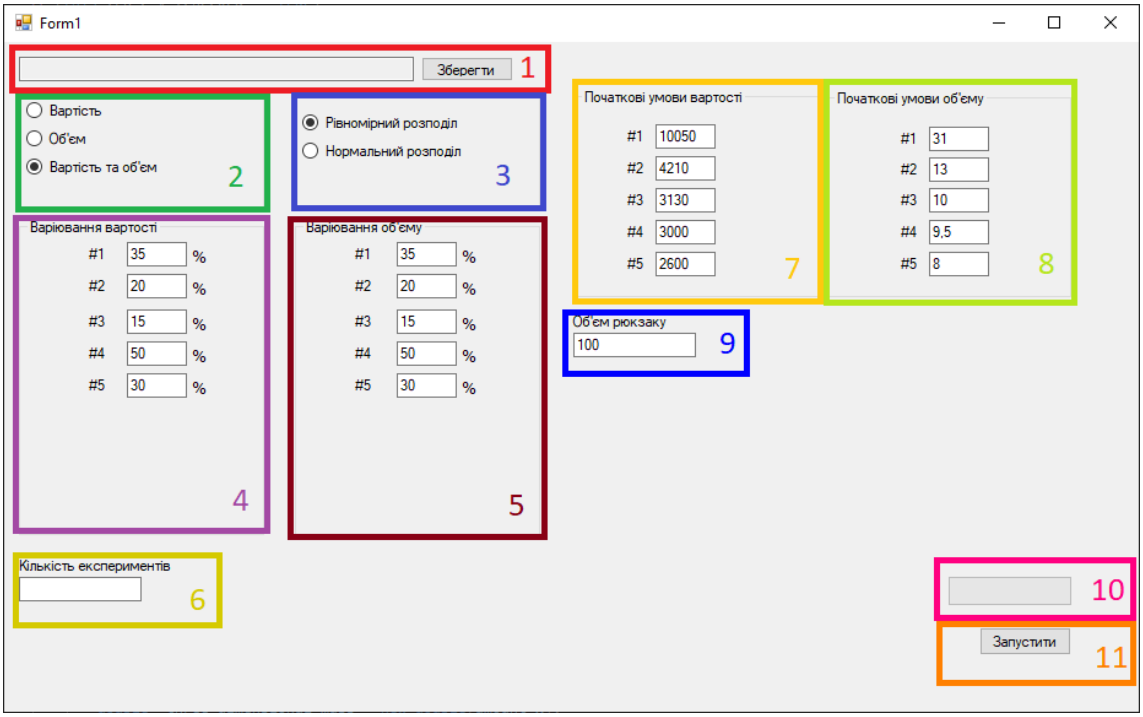

Рисунок 2.2 – Інструкція по інтерфейсу застосунка

Наведемо інструкцію інтерфейсу (рис 2.2):

Число «1»: Показує шлях до файлу після натискання кнопки «Зберегти». Кнопка «Зберегти» відповідає за збереження файлу і відкриває вікно з вашим

провідником, де потрібно вибрати шлях, де буде збережений файл з результатами.

Число «2»: В даному квадраті знаходяться перемикачі, які відповідають за різні варіювання в задачі про рюкзак такі, як «вартість», «об’єм» і «вартість та об’єм». Користувач їх може перемикати сам. За замовчування обрано «Вартість».

Число «3»: В даному квадраті знаходяться перемикачі, які відповідають за розподіли. В даному застосунку є два розподіли: нормальний та рівномірний. За замовчування обрано «Рівномірний розподіл».

Число «4»: Дана група об’єктів «Варіювання вартості» відповідає за відхилення вартості для кожного предмету. Є п’ять прямокутників для введення відхилення.

Число «5»: Дана група об’єктів «Варіювання об’єму» відповідає за відхилення об’єму для кожного предмету. Є п’ять прямокутників для введення відхилення.

Число «6»: В даному квадраті користувач може ввести кількість експериментів.

Число «7»: В даній групі об’єктів «Початкові умови вартості» можна ввести початкові умови кожного предмету.

Число «8»: В даній групі об’єктів «Початкові умови об’єм» можна ввести початкові умови кожного предмету.

Число «9»: В даному квадраті можна ввести максимальний об’єм рюкзаку за задачею.

Число «10»: В даному квадраті шкала, яка показує завершення роботи застосунку.

Число «11»: Кнопка «Запуска» роботи застосунку.

Після розроблення інтерфейсу потрібно було зв’язати інтерфейс з кодом програми, щоб введені дані були збережені застосунком та використовувалися в роботі.

Для повного функціонування програми розроблено такі методи та функції:

- функція для знаходження максимальної кількості упакованих предметів кожного типу;
- функція для знаходження кількості всіх можливих планів з урахуванням нижньої та верхньої межі кількості предметів кожного типу;
- функція, яка записує всі можливі плани в виділене місце або в створений масив, з урахуванням нижньої та верхньої межі кількості предметів кожного типу;
- метод повного перебору;
- функціонал для запису всіх даних в файл формату csv.

Завдяки оптимізації застосунку було вирішено:

- Проблему, коли користувач мав записувати окремо результат за кожним експериментом і перезапускати код програми по тисячу разів.
- Проблему, незручного змінення вхідних даних задачі.

Також було знайдено спеціальні бібліотеки в яких знаходяться функціонал для даної задачі, а саме нормальний та рівномірний розподіли. В роботі програми для рівномірного розподілу вказуються крайні точки, але в інтерфейсі написано варіювання у відсотках, тому було розроблено стандартну формулу для знаходження крайніх точок:

$$x_{min} = x - x * \left(\frac{\sigma}{100}\right), x_{max} = x + x * \left(\frac{\sigma}{100}\right),$$

де σ – відхилення у відсотках, x – початкове значення предмету.

Застосунок для рівномірного розподілу працює правильно завдяки проведення тестувань та підтвердженням тих самих результатів, що і в попередній роботі [13].

Новинкою для програми став нормальний розподіл. Щоб скористатися даним розподілом в функцію потрібно було ввести математичне сподівання та

середнє квадратичне відхилення, тобто σ (сігму). Але складність в тому, що потрібно підібрати таку сігму, яка показала б ті ж результати за нормаль ним розподілом, що і за рівномірний розподілом. І ще в програмі кожний предмет має своє відхилення, що ускладнює правильність підбору значення для даної ситуації. Тестуючи різні середні квадратичні відхилення програма не завжди записувала всі результати в файл. Це пов'язано з тим, що застосунок заздалегідь прораховує всі комбінації для даного випадку, а у випадку нормального розподілу значення може відхилятися аж на 3σ, де програма отримує результат, але порівнюючи з усіма комбінаціями в асортименті не знаходить його. Тому було обрано таку формулу для знаходження σ кожного предмету:

$$\sigma = (x * 0{,}2) * \left(\frac{\omega}{100}\right),$$

де ω – відхилення у відсотках, x – початкове значення предмету, σ – середнє квадратичне відхилення.

Дана формула є тестовим варіантом автора тому її використання в вашому випадку можливо не є доцільним.

Додам, що всі результати були протестовані і перевірені на доцільність і правильну роботу програми.

2.5 Критерії оптимальності задачі

В результаті розв’язку задачі можна отримати певну кількість оптимальних планів, яка може досягати декількох сотень, а необхідно обрати лише один оптимальний план. Виходячи з цього для отриманих оптимальних розв’язків потрібно застосувати певні критерії оптимальності.

Критерії оптимальності для задачі:

1 Критерій кількості з’явлень відповідного розв’язку у відсотках.

Для даного критерії було розроблено формулу для знаходження нижньої межі, за якою буде відбуватися відбір оптимальних планів. Формула виглядає таким чином:

$$l_{min} = \frac{m_{max}}{2},$$

де l_{min} – нижня межа для відбору оптимальних розв’язків;

m_{max} – максимальна кількість з’явлень серед оптимальних планів.

2 Критерій середньоарифметичної сумарної вартості відповідного розв’язку в у.о. (умовні одиниці).

Для даного критерії було розроблену формулу для знаходження сумарної вартості, за якою буде відбуватися відбір оптимальних планів. Формула виглядає таким чином:

$$q = \frac{(q_{max} + q_{min})}{2},$$

де q_{max} – максимальна середньоарифметична сумарна вартість серед оптимальних планів;

q_{min} – мінімальна середньоарифметична сумарна вартість серед оптимальних планів;

q – сумарна вартість для відбору оптимальних розв’язків.

Якщо за двома першими критеріями не можливо зробити так, щоб залишився один оптимальний план, то застосовуємо критерій ймовірності.

3 Критерій ймовірності, що отримана сумарна вартість для даного розв’язку була більше ніж k у.о.

Формула для даного критерію виглядає таким чином:

$$k = \frac{(q_{max} + q_{min})}{2},$$

де q_{max} – максимальна середньоарифметична сумарна вартість серед оптимальних планів;

q_{min} – мінімальна середньоарифметична сумарна вартість серед оптимальних планів;

k – сумарна вартість для відбору оптимальних розв’язків.

Для певного типу задачі цей критерій видозмінюється до вибору просто максимуму середньоарифметичної сумарної вартості серед відібраних після застосування першого та другого критеріїв оптимальних планів.

В такому вигляді задача є багатокритеріальною задачею оптимізації, яку пропонуємо розв'язувати зі застосуванням лексикографічного принципу.

3 АНАЛІЗ РЕЗУЛЬТАТІВ ЗАДАЧІ ПРО РЮКЗАК З РІВНОМІРНИМИ ТА НОРМАЛЬНИМИ ВХІДНИМИ ПАРАМЕТРАМИ

3.1 Задача 1

Розглянемо задачу пакування рюкзака, в якій вартість є випадковою величиною. Є 5 типів предметів, які потрібно упакувати в рюкзак. Об'єм рюкзаку дорівнює 100%. Предмет першого типу займає 31% об'єму рюкзаку, другого – 13%, третього – 10%, четвертого – 9,5% та п'ятого – 8%. Вартість одного предмету першого типу дорівнює 10050 у.о., другого – 4210 у.о., третього – 3130 у.о., четвертого – 3000 у.о. та п'ятого – 2600 у.о. Потрібно обов'язково упакувати не менше ніж по одному предмету першого, другого та третього типу.

Вартість для одного предмету кожного типу можуть варіюватися згідно певного закону розподілу відповідно наступних відсотків відхилення: один предмет першого типу – 35%, другого – 20%, третього – 15%, четвертого – 50% та п'ятого – 30%.

Необхідно упакувати рюкзак так, щоб сумарна вартість упакованих предметів була максимальною.

3.1.1 Аналіз оптимальних розв'язків задачі 1 з вхідним рівномірним параметром

Отримаємо розв'язки задачі, в якій вартість є випадковою величиною з рівномірним законом розподілу. Запустимо розроблений застосунок та заповнимо всі блоки вхідними параметрами і проведемо дослідження з кількістю експериментів в 20000 за рівномірним законом розподілу.

За результатами застосунку було отримано 8 оптимальних планів.

Скористаємося першим критерієм для відсіювання планів. Розраховуємо за формулою першого критерію l_{min} і отримаємо 17,85%. Після отримання нижньої межі відсіюємо ненадійні оптимальні плани.

Було відсіяно такі оптимальні плани: (1-1-4-0-2), (1-3-1-2-0), (1-3-3-0-0), (2-1-1-1-0), (2-1-2-0-0). В таблиці 3.1 наведені результати після застосування першого критерію.

Таблиця 3.1 – Результати за першим критерієм

Набір	Кількість
	20000
1 1 1 4 1	37,41%
1 2 1 0 4	20,05%
2 2 1 0 0	21,39%

Критерій середньоарифметичної сумарної вартості застосовується до результатів, що отримали після застосування критерія кількості з'явлень. Для початку розраховуємо за формулою другого критерію значення q і отримуємо 34634,8. Всі сумарні вартості, що нижче даного значення відсіюємо. В таблиці 3.2 наведені результати після застосування другого критерію.

Так як за двома критеріями не залишився один оптимальний план, то застосовуємо третій критерій, а саме критерій ймовірності, що отримана сумарна вартість для даного розв'язку була більше ніж k у.о. Розраховуємо k та отримаємо значення 35606,7. За цим значенням розраховуємо ймовірність для оптимальних розв'язків. В таблиці 3.3 наведені результати після застосування третього критерію.

Таблиця 3.2 – Результати за другим критерієм

Набір	Сер. арифм.
	20000
1 1 1 4 1	35209,7
2 2 1 0 0	36003,6

Таблиця 3.3 – Результати за третім критерієм

Набір	Ймовірність сум. варт. >35606,7
	20000
1 1 1 4 1	45,5
2 2 1 0 0	61,1

Після застосування третього критерія можна остаточно рекомендувати оптимальний план (2-2-1-0-0), а саме два предмети першого типу, два предмети другого типу та один предмет третього типу.

3.1.2 Аналіз оптимальних розв'язків задачі 1 з вхідним нормальним параметром

Отримаємо розв'язки задачі, в якій вартість є нормально розподіленою величиною. Запустимо розроблений застосунок та заповнимо всі блоки вхідними параметрами і проведемо дослідження з кількістю експериментів в 20000 за нормальним законом розподілу.

За результатами застосунку було отримано 6 оптимальних планів.

Скористаємося першим критерієм для відсіювання планів. Розраховуємо за формулою першого критерію l_{min} і отримаємо 17,85%. Після отримання нижньої межі відсіюємо ненадійні оптимальні плани. В таблиці 3.4 наведені результати після застосування першого критерію.

Після застосування першого критерія застосовуємо другий критерій. Розраховуємо за формулою другого критерію значення q і отримуємо 32631,8. Всі сумарні вартості, що нижче даного значення відсіюємо. В таблиці 3.5 наведені результати після застосування другого критерію.

Таблиця 3.4 – Результати за першим критерієм

Набір	Кількість
	20000
1 1 1 4 1	35,7%
1 2 1 0 4	20,8%
1 3 3 0 0	20,1%

Таблиця 3.5 – Результати за другим критерієм

Набір	Сер. арифм.
	20000
1 1 1 4 1	33073,4

Після застосування другого критерія можна остаточно рекомендувати оптимальний план (1-1-1-4-1), а саме один предмет першого типу, один предмет другого типу, один предмет третього типу, чотири предмета четвертого типу та один предмет п’ятого типу.

3.2 Задача 2

Розглянемо задачу пакування рюкзака, в якій об’єм є випадковою величиною. Є 5 типів предметів, які потрібно упакувати в рюкзак. Об’єм рюкзаку дорівнює 100%. Предмет першого типу займає 31% об’єму рюкзаку, другого – 13%, третього – 10%, четвертого – 9,5% та п’ятого – 8%. Вартість одного предмету першого типу дорівнює 10050 у.о., другого – 4210 у.о., третього – 3130 у.о., четвертого – 3000 у.о. та п’ятого – 2600 у.о. Потрібно обов’язково упакувати не менше ніж по одному предмету першого, другого та третього типу.

Об’єм для одного предмету кожного типу можуть варіюватися згідно певного закону розподілу відповідно наступних відсотків відхилення: один предмет першого типу – 35%, другого – 20%, третього – 15%, четвертого – 50% та п’ятого – 30%.

Необхідно упакувати рюкзак так, щоб сумарна вартість упакованих предметів була максимальною.

3.2.1 Аналіз оптимальних розв'язків задачі 2 з вхідним рівномірним параметром

Отримаємо розв'язки задачі, в якій об'єм є випадковою величиною з рівномірним законом розподілу. Запускаємо застосунок для рівномірного закону розподілу з 20000 кількістю експериментів. За результатами роботи була отримана така кількість оптимальних планів – 258.

Скористаємося першим критерієм для відсіювання планів. Розраховуємо за формулою першого критерію l_{min} і отримаємо 2,7%. Після отримання нижньої межі відсіюємо ненадійні оптимальні плани. В таблиці 3.6 наведені результати після застосування першого критерію.

Таблиця 3.6 – Результати за першим критерієм

Набір	Кількість
	20000
1 1 1 0 6	3,2
1 1 1 0 7	2,9
1 1 1 6 0	3,7
1 1 1 7 0	4,5
1 1 1 8 0	3,8
1 5 1 0 0	2,7
3 1 1 0 0	5,4
3 1 2 0 0	2,9

Після застосування першого критерія застосовуємо другий критерій. Розраховуємо за формулою другого критерію значення q і отримуємо 37190. Всі сумарні вартості, що нижче даного значення відсіюємо. В таблиці 3.7 наведені результати після застосування другого критерію.

Таблиця 3.7 – Результати за другим критерієм

Набір	Сер. арифм.
	20000
1 1 1 7 0	38390
1 1 1 8 0	41390
3 1 1 0 0	37490
3 1 2 0 0	40620

Побачивши результати можна сказати, що в даній задачі сумарна вартість певного плану не змінюється. Тому знайдемо оптимальний план з максимальною сумарною вартістю і будемо рекомендувати його. Це оптимальний план (1-1-1-8-0), а саме один предмет першого типу, один – другого, один – третього та вісім предметів четвертого типу.

3.2.2 Аналіз оптимальних розв'язків задачі 2 з вхідним нормальним параметром

Отримаємо розв'язки задачі, в якій об'єм є випадковою величиною з нормальним законом розподілу. Запускаємо застосунок для нормального закону розподілу з 20000 кількістю експериментів. За результатами роботи була отримана така кількість оптимальних планів – 116.

Скористаємося першим критерієм для відсіювання планів. Розраховуємо за формулою першого критерію l_{min} і отримаємо 3,3%. Після отримання нижньої межі відсіюємо ненадійні оптимальні плани. В таблиці 3.8 наведені результати після застосування першого критерію.

Таблиця 3.8 – Результати за першим критерієм

Набір	Кількість
	20000
1 1 1 0 6	6,6
1 1 1 5 0	3,4
1 1 2 4 0	3,9
1 2 1 0 4	6,6
1 2 1 4 0	3,7
1 3 1 0 2	3,8
1 3 3 0 0	3,4
2 1 1 0 2	4,3
2 1 2 0 1	3,7
2 1 3 0 0	3,3
2 2 1 0 1	3,3

Таблиця 3.9 – Результати за другим критерієм

Набір	Сер. арифм.
	20000
1 1 1 0 6	32990
1 2 1 4 0	33600
2 1 1 0 2	32640
2 1 2 0 1	33170
2 1 3 0 0	33700
2 2 1 0 1	34250

Після застосування першого критерія застосовуємо другий критерій. Розраховуємо за формулою другого критерію значення q і отримуємо 32630. Всі сумарні вартості, що нижче даного значення відсіюємо. В таблиці 3.9 наведені результати після застосування другого критерію.

Побачивши результати можна сказати, що в даній задачі сумарна вартість певного плану не змінюється. Тому знайдемо оптимальний план з максимальною сумарною вартістю і будемо рекомендувати його. Це оптимальний план (2-2-1-0-1), а саме два предмети першого типу, два – другого, один – третього та один предмет – п’ятого типу.

3.3 Задача 3

Розглянемо задачу, в якій об’єм та вартість утворюють систему незалежних випадкових величин. Є 5 типів предметів, які потрібно упакувати в рюкзак. Об’єм рюкзаку дорівнює 100%. Предмет першого типу займає 31% об’єму рюкзаку, другого – 13%, третього – 10%, четвертого – 9,5% та п’ятого – 8%. Вартість одного предмету першого типу дорівнює 10050 у.о., другого – 4210 у.о., третього – 3130 у.о., четвертого – 3000 у.о. та п’ятого – 2600 у.о. Потрібно обов’язково упакувати не менше ніж по одному предмету першого, другого та третього типу.

Вартість та об’єм для одного предмету кожного типу можуть варіюватися згідно певного закону розподілу відповідно наступних відсотків відхилення: один предмет першого типу – 35%, другого – 20%, третього – 15%, четвертого – 50% та п’ятого – 30%.

Необхідно упакувати рюкзак так, щоб сумарна вартість упакованих предметів була максимальною.

3.3.1 Аналіз оптимальних розв'язків задачі 3 з системою вхідних рівномірних параметрів

Отримаємо розв'язки задачі, в якій об'єм та вартість утворюють систему рівномірних випадкових величин. Запускаємо застосунок для рівномірного закону розподілу з 20000 кількістю експериментів. За результатами роботи була отримана така кількість оптимальних планів – 314.

Скористаємося першим критерієм для відсіювання планів. Розраховуємо за формулою першого критерію l_{min} і отримаємо 2,4%. Після отримання нижньої межі відсіюємо ненадійні оптимальні плани. В таблиці 3.10 наведені результати після застосування першого критерію.

Таблиця 3.10 – Результати за першим критерієм

Набір	Кількість
	20000
1 1 1 0 5	2,4
1 1 1 0 6	3,8
1 1 1 0 7	2,7
1 1 1 5 0	4,2
1 1 1 6 0	4,3
1 1 1 7 0	3,0
1 1 1 8 0	3,5
1 4 1 0 0	2,5
1 5 1 0 0	2,8
3 1 1 0 0	4,8

Після застосування першого критерія застосовуємо другий критерій. Розраховуємо за формулою другого критерію значення q і отримуємо 36419,45.

Всі сумарні вартості, що нижче даного значення відсіюємо. В таблиці 3.11 наведені результати після застосування другого критерію.

Таблиця 3.11 – Результати за другим критерієм

Набір	Сер. арифм.
	20000
1 1 1 6 0	39561,1
1 1 1 7 0	42178,5
1 1 1 8 0	45267,8
3 1 1 0 0	41981,7

Так як за двома критеріями не залишився один оптимальний план, то застосовуємо третій критерій, а саме критерій ймовірності, що отримана сумарна вартість для даного розв'язку була більше ніж *k* у.о. Розраховуємо *k* та отримаємо значення 42414,45. За цим значення розраховуємо ймовірність для оптимальних розв'язків. В таблиці 3.12 наведені результати після застосування третього критерію.

Таблиця 3.12 – Результати за третім критерієм

Набір	Ймовірність сум варт >42414,45
	20000
1 1 1 6 0	25,4
1 1 1 7 0	51,3
1 1 1 8 0	68,7
3 1 1 0 0	52,0

Після застосування третього критерія можна остаточно рекомендувати оптимальний план (1-1-1-8-0), а саме один предмет першого типу, один предмет другого типу, один предмет третього типу та вісім предметів четвертого типу.

3.3.2 Аналіз оптимальних розв'язків задачі 3 з системою вхідних нормальних параметрів

Отримаємо розв'язки задачі, в якій об'єм та вартість утворюють систему нормальних випадкових величин. Запускаємо застосунок для рівномірного закону розподілу з 20000 кількістю експериментів. За результатами роботи була отримана така кількість оптимальних планів – 116.

Скористаємося першим критерієм для відсіювання планів. Розраховуємо за формулою першого критерію l_{min} і отримаємо 3,5%. Після отримання нижньої межі відсіюємо ненадійні оптимальні плани. В таблиці 3.13 наведені результати після застосування першого критерію.

Таблиця 3.13 – Результати за першим критерієм

Набір	Кількість, відсоток
	20000
1 1 1 0 6	6,1
1 1 1 5 0	7,0
1 2 1 0 4	5,2
1 4 1 0 1	3,6
2 1 1 0 2	3,9
2 2 1 0 0	4,6

Після застосування першого критерія застосовуємо другий критерій. Розраховуємо за формулою другого критерію значення q і отримуємо 33147,45. Всі сумарні вартості, що нижче даного значення відсіюємо. В таблиці 3.14 наведені результати після застосування другого критерію.

Таблиця 3.14 – Результати за другим критерієм

Набір	Сер. арифм.
	20000
1 1 1 0 6	33415,9
1 1 1 5 0	33733,9
2 1 1 0 2	33593,4

Так як за двома критеріями не залишився один оптимальний план, то застосовуємо третій критерій, а саме критерій ймовірності, що отримана сумарна вартість для даного розв'язку була більше ніж k у.о. Розраховуємо k та отримаємо значення 33592,9. За цим значення розраховуємо ймовірність для оптимальних розв'язків. В таблиці 3.15 наведені результати після застосування третього критерію.

Таблиця 3.15 – Результати за третім критерієм

Набір	Ймовірність сум варт >33592,9
	20000
1 1 1 0 6	44,3
1 1 1 5 0	53,3
2 1 1 0 2	49,2

Після застосування третього критерія можна остаточно рекомендувати оптимальний план (1-1-1-5-0), а саме один предмет першого типу, один предмет другого типу, один предмет третього типу та п’ять предметів четвертого типу.

ВИСНОВКИ

В дипломні роботі виконано такі завдання:

- Описано підхід знаходження оптимального розв’язку задачі, вхідні параметри якої є системою незалежних випадкових величин.
- Розроблено застосунок з інтерфейсом, який знаходить оптимальні розв’язки та сумарну вартість і записує результати в файл з заданою кількістю експериментів. Інтерфейс застосунку дозволяє обирати задачу та закон розподілу, задавати початкові умови вартості та об’єму, максимальний об’єм рюкзака та відсоток відхилення вартості або/та об’єму для кожного предмету.
- Розроблено критерії оптимальності для аналізу впливу невизначеності на оптимальний план задачі.
- Отримано та проаналізовано оптимальні розв’язки першої задачі з варіюванням вартості, другої задачі з варіюванням об’єму і третьої задачі з варіюванням вартості та об’єму згідно нормальному та рівномірному законам розподілу.
- Дослідження вплив статистичної невизначеності на прикладі рівномірного та нормального законів розподілу на оптимальний розв’язок задачі про рюкзак.

ПЕРЕЛІК ПОСИЛАНЬ

1. **Березовий М.І.** Визначення оптимального варіанту розподілу сортувальних колій між призначеннями вагонопотоків [Текст] / М. І. Березовий (Дніпропетровський національний університет залізничного транспорту імені академіка В. Лазаряна), С. О. Пожидаєв (Білоруський держаний університет транспорту, м. Гомель, Республіка Бєларусь), В. С. Буров (Кримська філія МДА, Севастополь) // Транспортні системи та технології перевезень. – Дніпровський національний університет залізничного транспорту імені академіка В. Лазаряна, Дніпро, 2011. – с. 5–9.

2. **Концеба С.М.** Економіко-математична модель оптимізації функціонування плодоовочесховищ [Текст] / С. М. Концеба кандидат економічних наук, старший викладач, кафедра економічної кібернетики та інформаційних систем. // БІЗНЕСІНФОРМ № 2. – Умань: Уманський національний університет садівництва, 2013. – с. 79–82.

3. **Медиковський М.О.** Застосування динамічного програмування для задачі рівномірного використання вітрових електроустановок [Текст] / М. О. Медиковський, докт. техн. наук, В. М. Теслюк, докт.техн.наук, О. Б. Шуневич, канд. техн. наук. // Техн. електродинаміка. №4. – Львів : НУЛП, 2014. – с.135–137.

4. **Маслова Н.О.** Використання інтелектуальних агентів при рішенні задач розподілу ресурсів [Текст] / Н.О. Маслова, О.В. Мовчан // «Штучний інтелект», 2014, № 3 – Інститут проблем штучного інтелекту Міністерства освіти і науки України і Національної академії наук України, Київ, 2014. – с. 80–89.

5. **Яковлева А.П.** Дослідження задач знаходження оптимального розподілу ресурсів між підприємствами [Текст] / А.П. Яковлева, І.О. Курдуп // Системні дослідження та інформаційні технології, 2016, №2. – Інститут управління системного аналізу України, Київ, 2016. – с. 74–80.

6. **Кравченко Є.І.** Алгоритм гілок та меж для задачі формування команди та пошуку інвесторів для стартап-проектів [Текст] / Кравченко Є. І. // Журнал науковий огляд – 2018 р. – №6(49) – с. 70 – 76.

7. **Ткачук В.М.** Квантовий генетичний алгоритм в задачі 0-1 пакування мультиплікативного рюкзака [Текст] / // Інформаційні технології та взаємодії – 2018 р. – с. 220 – 221.

8. **Зленко Ю.** Математична складова задач динамічного програмування [Текст] / Ю. Зленко ; наук. кер. Л. В. Ізюмченко // Фізико-математичні та комп'ютерні науки, технології, навчання: науково-практичні рішення та підходи молодих науковців. – Кропивницький : ЦДПУ ім. В. Винниченка , 2018. – с. 1–11.

9. **Качор П.М.** Застосування задачі пакування рюкзака для оптимального розподілення ресурсів-сервера [Текст] / П. М. Качор ; наук. кер. Н. О. Бондаренко // Мехатронні системи і комп'ютерні технології Т.2: Ресурсозбереження та охорона навколишнього середовища. – Київ : КНУТД, 2019. – с. 67–68.

10. **Рибачок І.А.** Упакування куль у опуклому багатограннику із зонами заборони [Текст]: диплом магістр / І.А. Рибачок – Харків, 2020. – 58 с.

11. **Безсонний В.Л.** Розробка алгоритму оптимального управління ризиками небезпечних подій на машинобудівному підприємстві [Текст] / В. Л. Безсонний , к.т.н., доцент, доц. каф., Р. В. Пономаренко , д.т.н., с.н.с., заст. нач. каф., О. В. Третьяков , д.т.н., професор, наук. консультант, К. М. Карпець , к.геогр.н., доцент, провідн. н.с. // Проблеми надзвичайних ситуацій. 2021. № 1(33). – Харьків : ХНЕУ ім. С. Кузнеця, 2021. – с.58–71.

12. **Галісєєв Г.В.** Системне програмування [Текст] : навч. посіб. / Галісеєв Г. В. - Київ : Ун-т "Україна", 2019. – 112 с. : іл., рис., табл. – 50 прим.

13. **Смола В.І.** Дослідження задачі про рюкзак в умовах невизначеності [Текст]: диплом бакалавр / В.І. Смола – Запоріжжя, 2021. – 48 с.

ДОДАТОК А

Код C++

```
#pragma once
#include<string>
#include<random>
#include<fstream>
#include<sstream>
#include<string>
#include<direct.h>
…
private: System::Void button1_Click(System::Object^ sender, System::EventArgs^ e) {
if (textBox1->Text != ""){
this->progressBar3->Value = 0;
std::string path;
MarshalString(textBox1->Text, path);
int path_amount = path.length();
char* path_h = new char[path_amount + 1];
strcpy(path_h, path.c_str());
mkdir(path_h);
if (this->rB_c->Checked == true){
if ((tB_c_1->Text == "0" || tB_c_2->Text == "0" || tB_c_3->Text == "0"
&& tB_c_4->Text == "0" || tB_c_5->Text == "0") && this->rB_norm_dist->Checked == true) {
MessageBox::Show("Error! \nSigma > 0 in normal distribution");}
else {
double* interes = new double[5];
interes[0] = Convert::ToDouble(tB_c_1->Text) / 100;
interes[1] = Convert::ToDouble(tB_c_2->Text) / 100;
interes[2] = Convert::ToDouble(tB_c_3->Text) / 100;
interes[3] = Convert::ToDouble(tB_c_4->Text) / 100;
interes[4] = Convert::ToDouble(tB_c_5->Text) / 100;
int* amount = new int[1];
amount[0] = Convert::ToInt32(tB_count_res_file->Text);
int x_amount = 5;

double *x_volume = new double[5];
x_volume[0] = Convert::ToDouble(tB_pu_v_1->Text);
x_volume[1] = Convert::ToDouble(tB_pu_v_2->Text);
x_volume[2] = Convert::ToDouble(tB_pu_v_3->Text);
x_volume[3] = Convert::ToDouble(tB_pu_v_4->Text);
x_volume[4] = Convert::ToDouble(tB_pu_v_5->Text);
double* x_cost = new double[5];
```

```
x_cost[0] = Convert::ToDouble(tB_pu_c_1->Text);
x_cost[1] = Convert::ToDouble(tB_pu_c_2->Text);
x_cost[2] = Convert::ToDouble(tB_pu_c_3->Text);
x_cost[3] = Convert::ToDouble(tB_pu_c_4->Text);
x_cost[4] = Convert::ToDouble(tB_pu_c_5->Text);
double max_volume_x = Convert::ToDouble(tB_v_backpack->Text);

double* three_dimensional_mass = new double[x_amount];
int* x_max = new int[5];
x_max[0] = 3;
x_max[1] = 7;
x_max[2] = 10;
x_max[3] = 10;
x_max[4] = 12;
int* x_min = new int[5];
x_min[0] = 1;
x_min[1] = 1;
x_min[2] = 1;
x_min[3] = 0;
x_min[4] = 0;
int all_combination = Amount_Combo(x_max, x_min, x_volume, x_amount, max_volume_x);
int** amount_mass = new int* [all_combination];
for (int i = 0; i < all_combination; i++) {
amount_mass[i] = new int[x_amount];
}
All_Combo(x_max, x_min, x_volume, x_amount, max_volume_x, amount_mass);
int* x = new int[x_amount];
int* x_opt = new int[x_amount];
double f, f_opt;
int* mass_int_combi = new int[all_combination];
int counter_while, _j;
double _max_volume_x;
int count_while = 0;
int count_right = 0;
while (count_while < 1) {
double** mass = new double* [amount[count_while]];
for (int i = 0; i < amount[count_while]; i++)
mass[i] = new double[all_combination];
for (int i = 0; i < all_combination; i++)
mass_int_combi[i] = 0;
counter_while = 0;
while (counter_while < amount[count_while]) {
std::random_device rd;
std::mt19937 gen(rd());
if (this->rB_unif_dist->Checked == true) {
```

```
        std::uniform_real_distribution<> rndC1(x_cost[0] - x_cost[0] * interes[0], x_cost[0] +
x_cost[0] * interes[0]);
        three_dimensional_mass[0] = rndC1(gen);
        std::uniform_real_distribution<> rndC2(x_cost[1] - x_cost[1] * interes[1], x_cost[1] +
x_cost[1] * interes[1]);
        three_dimensional_mass[1] = rndC2(gen);
        std::uniform_real_distribution<> rndC3(x_cost[2] - x_cost[2] * interes[2], x_cost[3] +
x_cost[3] * interes[2]);
        three_dimensional_mass[2] = rndC3(gen);
        std::uniform_real_distribution<> rndC4(x_cost[3] - x_cost[3] * interes[3], x_cost[3] +
x_cost[3] * interes[3]);
        three_dimensional_mass[3] = rndC4(gen);
        std::uniform_real_distribution<> rndC5(x_cost[4] - x_cost[4] * interes[4], x_cost[4] +
x_cost[4] * interes[4]);
        three_dimensional_mass[4] = rndC5(gen);
        }
        else {
        std::normal_distribution<> rndC11(x_cost[0], (x_cost[0] * 0.1 * 2) * interes[0]);
        three_dimensional_mass[0] = rndC11(gen);
        std::normal_distribution<> rndC22(x_cost[1], (x_cost[1] * 0.1 * 2) * interes[1]);
        three_dimensional_mass[1] = rndC22(gen);
        std::normal_distribution<> rndC33(x_cost[2], (x_cost[2] * 0.1 * 2) * interes[2]);
        three_dimensional_mass[2] = rndC33(gen);
        std::normal_distribution<> rndC44(x_cost[3], (x_cost[3] * 0.1 * 2) * interes[3]);
        three_dimensional_mass[3] = rndC44(gen);
        std::normal_distribution<> rndC55(x_cost[4], (x_cost[4] * 0.1 * 2) * interes[4]);
        three_dimensional_mass[4] = rndC55(gen);
        }
        for (int i = 0; i < x_amount; i++)
        x_opt[i] = x_min[i];
        for (int i = 0; i < x_amount; i++)
        x[i] = x_min[i];
        f = 0;
        f_opt = -9999;
        _j = x_amount - 1;
        _max_volume_x = 0;
        while (true) {
        if (x[_j] <= x_max[_j]) {
        for (int i = 0; i < x_amount; i++)
        _max_volume_x += x_volume[i] * x[i];
        if (_max_volume_x <= max_volume_x) {
        for (int i = 0; i < x_amount; i++)
        f += three_dimensional_mass[i] * x[i];
        if (f_opt <= f) {
        f_opt = f;
```

```
for (int i = 0; i < x_amount; i++)
x_opt[i] = x[i];
}
}
_j = x_amount - 1;
}
else {
x[_j] = x_min[_j];
_j = _j - 1;
if (_j < 0)
break;
}
x[_j] += 1;
_max_volume_x = 0;
f = 0;
}
count_right = 0;
for (int i = 0; i < all_combination; i++) {
for (int k = 0; k < x_amount; k++)
if (amount_mass[i][k] == x_opt[k])
count_right++;
if (count_right == x_amount) {
mass[mass_int_combi[i]][i] = f_opt;
mass_int_combi[i]++;
count_right = 0;
}
else count_right = 0;
}
counter_while++;
}
int max = 0;
for (int i = 0; i < all_combination; i++)
if (max < mass_int_combi[i])
max = mass_int_combi[i];
std::string* gyper = new std::string[1];
if    (this->rB_unif_dist->Checked    ==    true){gyper[0]    =    path    +    "/"    +
std::to_string(amount[0]) + " uniform cost.csv";}
else{gyper[0] = path + "/" + std::to_string(amount[0]) + " normal cost.csv";}
std::ofstream csvFileResult(gyper[count_while]);
for (int i = 0; i < all_combination; i++) {
if (mass_int_combi[i] != 0) {
csvFileResult << "_";
for (int k = 0; k < x_amount; k++)
csvFileResult << amount_mass[i][k];
if (i != all_combination - 1)
```

```
        csvFileResult << "_,";
        }
        if (i == all_combination - 1)
        csvFileResult << "_";
        }
        csvFileResult << "\n";
        double* s_arif_sum = new double[all_combination];
        for (int i = 0; i < all_combination; i++)
        s_arif_sum[i] = 0;
        for (int i = 0; i < max; i++) {
        for (int mk = 0; mk < all_combination; mk++) {
        if (mass_int_combi[mk] != 0) {
        if (mass_int_combi[mk] > i) {
        s_arif_sum[mk] += mass[i][mk];
        csvFileResult << mass[i][mk];
        }
        if (mk != all_combination - 1)
        csvFileResult << ",";
        }
        if (mk == all_combination - 1)
        csvFileResult << "\n";
        }
        }
        for (int i = 0; i < all_combination; i++) {
        if (mass_int_combi[i] != 0) {
        csvFileResult << "_";
        for (int k = 0; k < x_amount; k++)
        csvFileResult << amount_mass[i][k];
        if (i != all_combination - 1)
        csvFileResult << "_,";
        }
        if (i == all_combination - 1)
        csvFileResult << "_";
        }
        csvFileResult << "\n";

        for (int mk = 0; mk < all_combination; mk++) {
        if (mass_int_combi[mk] != 0) {
        csvFileResult << Convert::ToDouble(mass_int_combi[mk]) / Convert::ToDouble(amount[0]) *
100;
        if (mk != all_combination - 1)
        csvFileResult << ",";
        }
        if (mk == all_combination - 1)
        csvFileResult << ",Кількість з'явлень\n";
```

```
}

for (int mk = 0; mk < all_combination; mk++) {
if (mass_int_combi[mk] != 0) {
csvFileResult << (s_arif_sum[mk] / Convert::ToDouble(mass_int_combi[mk]));
if (mk != all_combination - 1)
csvFileResult << ",";
}
if (mk == all_combination - 1)
csvFileResult << ",Середнєариф\n";
}

double counters_;
for (int mk = 0; mk < all_combination; mk++) {
if (mass_int_combi[mk] != 0) {
counters_ = 0;
for (int i = 0; i < max; i++)
if (mass_int_combi[mk] > i)
if (mass[i][mk] > 30000)
counters_ += 1;
csvFileResult << (counters_ / Convert::ToDouble(mass_int_combi[mk])) * 100;
if (mk != all_combination - 1)
csvFileResult << ",";
}
if (mk == all_combination - 1)
csvFileResult << ",Імовірність >30000\n";
}

for (int mk = 0; mk < all_combination; mk++) {
if (mass_int_combi[mk] != 0) {
counters_ = 0;
for (int i = 0; i < max; i++)
if (mass_int_combi[mk] > i)
if (mass[i][mk] > 35000)
counters_ += 1;
csvFileResult << (counters_ / Convert::ToDouble(mass_int_combi[mk])) * 100;
if (mk != all_combination - 1)
csvFileResult << ",";
}
if (mk == all_combination - 1)
csvFileResult << ",Імовірність >35000\n";
}
csvFileResult.close();
count_while++;
}
```

```
}
}
if (this->rB_v->Checked == true) {
if ((tB_v_1->Text == "0" || tB_v_2->Text == "0" || tB_v_3->Text == "0"
&& tB_v_4->Text == "0" || tB_v_5->Text == "0") && this->rB_norm_dist->Checked == true){
MessageBox::Show("Error! \nSigma > 0 in normal distribution");}
else{
double* interes = new double[5];
interes[0] = Convert::ToDouble(tB_v_1->Text) / 100;
interes[1] = Convert::ToDouble(tB_v_2->Text) / 100;
interes[2] = Convert::ToDouble(tB_v_3->Text) / 100;
interes[3] = Convert::ToDouble(tB_v_4->Text) / 100;
interes[4] = Convert::ToDouble(tB_v_5->Text) / 100;
int *amount = new int[1];
amount[0] = Convert::ToInt32(tB_count_res_file->Text);
int x_amount = 5;
double* x_volume = new double[5];
x_volume[0] = Convert::ToDouble(tB_pu_v_1->Text);
x_volume[1] = Convert::ToDouble(tB_pu_v_2->Text);
x_volume[2] = Convert::ToDouble(tB_pu_v_3->Text);
x_volume[3] = Convert::ToDouble(tB_pu_v_4->Text);
x_volume[4] = Convert::ToDouble(tB_pu_v_5->Text);
double* x_cost = new double[5];
x_cost[0] = Convert::ToDouble(tB_pu_c_1->Text);
x_cost[1] = Convert::ToDouble(tB_pu_c_2->Text);
x_cost[2] = Convert::ToDouble(tB_pu_c_3->Text);
x_cost[3] = Convert::ToDouble(tB_pu_c_4->Text);
x_cost[4] = Convert::ToDouble(tB_pu_c_5->Text);
double max_volume_x = Convert::ToDouble(tB_v_backpack->Text);
double* three_dimensional_mass = new double[x_amount];
int* x_max = new int[x_amount];
x_max[0] = max_volume_x / (x_volume[0] - x_volume[0] * interes[0]);
x_max[1] = max_volume_x / (x_volume[1] - x_volume[1] * interes[1]);
x_max[2] = max_volume_x / (x_volume[2] - x_volume[2] * interes[2]);
x_max[3] = max_volume_x / (x_volume[3] - x_volume[3] * interes[3]);
x_max[4] = max_volume_x / (x_volume[4] - x_volume[4] * interes[4]);
int x_min[5] = { 1,1,1,0,0 };
double _x_volume[5];
_x_volume[0] = x_volume[0] - x_volume[0] * interes[0];
_x_volume[1] = x_volume[1] - x_volume[1] * interes[1];
_x_volume[2] = x_volume[2] - x_volume[2] * interes[2];
_x_volume[3] = x_volume[3] - x_volume[3] * interes[3];
_x_volume[4] = x_volume[4] - x_volume[4] * interes[4];
int all_combination = Amount_Combo(x_max, x_min, _x_volume, x_amount, max_volume_x);
int** count_sample = new int* [all_combination];
```

```
        for (int i = 0; i < all_combination; i++) {
        count_sample[i] = new int[x_amount];
        }
        All_Combo(x_max, x_min, _x_volume, x_amount, max_volume_x, count_sample);
        int* x = new int[x_amount];
        int* x_opt = new int[x_amount];
        double f, f_opt;
        int* mass_int_combi = new int[all_combination];
        int counter_while, _j;
        double _max_volume_x;
        int count_while = 0;
        int count_right = 0;
        while (count_while < 1) {
        double** mass = new double* [amount[count_while]];
        for (int i = 0; i < amount[count_while]; i++)
        mass[i] = new double[all_combination];
        for (int i = 0; i < all_combination; i++)
        mass_int_combi[i] = 0;
        counter_while = 0;
        while (counter_while < amount[count_while]) {
        std::random_device rd;
        std::mt19937 gen(rd());
        if (this->rB_unif_dist->Checked == true){
        std::uniform_real_distribution<>   rndC1(x_volume[0]   -   x_volume[0]   *   interes[0],
x_volume[0] + x_volume[0] * interes[0]);
        three_dimensional_mass[0] = rndC1(gen);
        std::uniform_real_distribution<>   rndC2(x_volume[1]   -   x_volume[1]   *   interes[1],
x_volume[1] + x_volume[1] * interes[1]);
        three_dimensional_mass[1] = rndC2(gen);
        std::uniform_real_distribution<>   rndC3(x_volume[2]   -   x_volume[2]   *   interes[2],
x_volume[2] + x_volume[2] * interes[2]);
        three_dimensional_mass[2] = rndC3(gen);
        std::uniform_real_distribution<>   rndC4(x_volume[3]   -   x_volume[3]   *   interes[3],
x_volume[3] + x_volume[3] * interes[3]);
        three_dimensional_mass[3] = rndC4(gen);
        std::uniform_real_distribution<>   rndC5(x_volume[4]   -   x_volume[4]   *   interes[4],
x_volume[4] + x_volume[4] * interes[4]);
        three_dimensional_mass[4] = rndC5(gen);}
        else{
        std::normal_distribution<> rndC1(x_volume[0], (x_volume[0] * 0.1 * 2) * interes[0]);
        three_dimensional_mass[0] = rndC1(gen);
        std::normal_distribution<> rndC2(x_volume[1], (x_volume[1] * 0.1 * 2)* interes[1]);
        three_dimensional_mass[1] = rndC2(gen);
        std::normal_distribution<> rndC3(x_volume[2], (x_volume[2] * 0.1 * 2)* interes[2]);
        three_dimensional_mass[2] = rndC3(gen);
```

```
std::normal_distribution<> rndC4(x_volume[3], (x_volume[3] * 0.1 * 2)* interes[3]);
three_dimensional_mass[3] = rndC4(gen);
std::normal_distribution<> rndC5(x_volume[4], (x_volume[4] * 0.1 * 2) * interes[4]);
three_dimensional_mass[4] = rndC5(gen);}
for (int i = 0; i < x_amount; i++)
x_opt[i] = x_min[i];
for (int i = 0; i < x_amount; i++)
x[i] = x_min[i];
f = 0;
f_opt = -9999;
_j = x_amount - 1;
_max_volume_x = 0;
while (true) {
if (x[_j] <= x_max[_j]) {
for (int i = 0; i < x_amount; i++)
_max_volume_x += three_dimensional_mass[i] * x[i];
if (_max_volume_x <= max_volume_x) {
for (int i = 0; i < x_amount; i++)
f += x_cost[i] * x[i];
if (f_opt <= f) {
f_opt = f;
for (int i = 0; i < x_amount; i++)
x_opt[i] = x[i];
}
}
_j = x_amount - 1;
}
else {
x[_j] = x_min[_j];
_j = _j - 1;
if (_j < 0)
break;
}
x[_j] += 1;
_max_volume_x = 0;
f = 0;
}
count_right = 0;
for (int i = 0; i < all_combination; i++) {
for (int k = 0; k < x_amount; k++)
if (count_sample[i][k] == x_opt[k])
count_right++;
if (count_right == x_amount) {
mass[mass_int_combi[i]][i] = f_opt;
mass_int_combi[i]++;
```

```
count_right = 0;
}
else count_right = 0;
}
counter_while++;
}
int max = 0;
int prokol = 0;
for (int i = 0; i < all_combination; i++) {
if (max < mass_int_combi[i])
max = mass_int_combi[i];
prokol += mass_int_combi[i];
}
std::string* gyper = new std::string[1];
if (this->rB_unif_dist->Checked == true)
gyper[0] = path + "/" + std::to_string(amount[0]) + " uniform volume.csv";
else
gyper[0] = path + "/" + std::to_string(amount[0]) + " normal volume.csv";
int count_plan = 0;
std::ofstream csvFileResult(gyper[count_while]);
for (int i = 0; i < all_combination; i++) {
if (mass_int_combi[i] != 0) {
count_plan++;
csvFileResult << "_";
for (int k = 0; k < x_amount; k++)
csvFileResult << count_sample[i][k];
if (i != all_combination - 1)
csvFileResult << "_,";
}
if (i == all_combination - 1)
csvFileResult << "_";
}
csvFileResult << "\n";
double* s_arif_sum = new double[all_combination];
for (int i = 0; i < all_combination; i++)
s_arif_sum[i] = 0;
for (int i = 0; i < max; i++) {
for (int mk = 0; mk < all_combination; mk++) {
if (mass_int_combi[mk] != 0) {
if (mass_int_combi[mk] > i) {
s_arif_sum[mk] += mass[i][mk];
csvFileResult << mass[i][mk];
}
if (mk != all_combination - 1)
csvFileResult << ",";
```

```
        }
        if (mk == all_combination - 1)
        csvFileResult << "\n";
        }
        }

        for (int i = 0; i < all_combination; i++) {
        if (mass_int_combi[i] != 0) {
        csvFileResult << "_";
        for (int k = 0; k < x_amount; k++)
        csvFileResult << count_sample[i][k];
        if (i != all_combination - 1)
        csvFileResult << "_,";
        }
        if (i == all_combination - 1)
        csvFileResult << "_";
        }
        csvFileResult << "\n";
        for (int mk = 0; mk < all_combination; mk++) {
        if (mass_int_combi[mk] != 0) {
        csvFileResult << Convert::ToDouble(mass_int_combi[mk]) / Convert::ToDouble(amount[0]) *
100;
        if (mk != all_combination - 1)
        csvFileResult << ",";
        }
        if (mk == all_combination - 1)
        csvFileResult << ",Кількість з'явлень\n";
        }
        for (int mk = 0; mk < all_combination; mk++) {
        if (mass_int_combi[mk] != 0) {
        csvFileResult << (s_arif_sum[mk] / Convert::ToDouble(mass_int_combi[mk]));
        if (mk != all_combination - 1)
        csvFileResult << ",";
        }
        if (mk == all_combination - 1)
        csvFileResult << ",Середнєариф\n";
        }
        double counters_;
        for (int mk = 0; mk < all_combination; mk++) {
        if (mass_int_combi[mk] != 0) {
        counters_ = 0;
        for (int i = 0; i < max; i++)
        if (mass_int_combi[mk] > i)
        if (mass[i][mk] > 30000)
        counters_ += 1;
```

```
csvFileResult << (counters_ / Convert::ToDouble(mass_int_combi[mk])) * 100;
if (mk != all_combination - 1)
csvFileResult << ",";
}
if (mk == all_combination - 1)
csvFileResult << ",Імовірність >30000\n";
}
for (int mk = 0; mk < all_combination; mk++) {
if (mass_int_combi[mk] != 0) {
counters_ = 0;
for (int i = 0; i < max; i++)
if (mass_int_combi[mk] > i)
if (mass[i][mk] > 35000)
counters_ += 1;
csvFileResult << (counters_ / Convert::ToDouble(mass_int_combi[mk])) * 100;
if (mk != all_combination - 1)
csvFileResult << ",";
}
if (mk == all_combination - 1)
csvFileResult << ",Імовірність >35000\n";
}
csvFileResult.close();
count_while++;
}
}
}
if (this->rB_c_v->Checked == true)
{
if ((tB_c_1->Text == "0" || tB_c_2->Text == "0" || tB_c_3->Text == "0"
|| tB_c_4->Text == "0" || tB_c_5->Text == "0" || tB_v_1->Text == "0"
|| tB_v_2->Text == "0" || tB_v_3->Text == "0"|| tB_v_4->Text == "0"
|| tB_v_5->Text == "0") && this->rB_norm_dist->Checked == true){
MessageBox::Show("Error! \nSigma > 0 in normal distribution");}
else{
double* interes = new double[10];
interes[0] = Convert::ToDouble(tB_c_1->Text) / 100;
interes[1] = Convert::ToDouble(tB_c_2->Text) / 100;
interes[2] = Convert::ToDouble(tB_c_3->Text) / 100;
interes[3] = Convert::ToDouble(tB_c_4->Text) / 100;
interes[4] = Convert::ToDouble(tB_c_5->Text) / 100;
interes[5] = Convert::ToDouble(tB_v_1->Text) / 100;
interes[6] = Convert::ToDouble(tB_v_2->Text) / 100;
interes[7] = Convert::ToDouble(tB_v_3->Text) / 100;
interes[8] = Convert::ToDouble(tB_v_4->Text) / 100;
interes[9] = Convert::ToDouble(tB_v_5->Text) / 100;
```

```
//int amount[3] = { 1000,5000,10000 };
int* amount = new int[1];
amount[0] = Convert::ToInt32(tB_count_res_file->Text);
int amount_x = 5;
double* volume_x = new double[5];
volume_x[0] = Convert::ToDouble(tB_pu_v_1->Text);
volume_x[1] = Convert::ToDouble(tB_pu_v_2->Text);
volume_x[2] = Convert::ToDouble(tB_pu_v_3->Text);
volume_x[3] = Convert::ToDouble(tB_pu_v_4->Text);
volume_x[4] = Convert::ToDouble(tB_pu_v_5->Text);
double* x_cost = new double[5];
x_cost[0] = Convert::ToDouble(tB_pu_c_1->Text);
x_cost[1] = Convert::ToDouble(tB_pu_c_2->Text);
x_cost[2] = Convert::ToDouble(tB_pu_c_3->Text);
x_cost[3] = Convert::ToDouble(tB_pu_c_4->Text);
x_cost[4] = Convert::ToDouble(tB_pu_c_5->Text);
double max_volume_x = Convert::ToDouble(tB_v_backpack->Text);
double* three_dimensional_mass = new double[amount_x];
double* three_dimensional_mass_1 = new double[amount_x];
int* x_max = new int[amount_x];
x_max[0] = max_volume_x / (volume_x[0] - volume_x[0] * interes[5]);
x_max[1] = max_volume_x / (volume_x[1] - volume_x[1] * interes[6]);
x_max[2] = max_volume_x / (volume_x[2] - volume_x[2] * interes[7]);
x_max[3] = max_volume_x / (volume_x[3] - volume_x[3] * interes[8]);
x_max[4] = max_volume_x / (volume_x[4] - volume_x[4] * interes[9]);
int x_min[5] = { 1,1,1,0,0 };
double _x_volume[5];
_x_volume[0] = volume_x[0] - volume_x[0] * interes[5];
_x_volume[1] = volume_x[1] - volume_x[1] * interes[6];
_x_volume[2] = volume_x[2] - volume_x[2] * interes[7];
_x_volume[3] = volume_x[3] - volume_x[3] * interes[8];
_x_volume[4] = volume_x[4] - volume_x[4] * interes[9];
int all_combination = Amount_Combo(x_max, x_min, _x_volume, amount_x, max_volume_x);
int** count_sample = new int* [all_combination];
for (int i = 0; i < all_combination; i++)
count_sample[i] = new int[amount_x];
All_Combo(x_max, x_min, _x_volume, amount_x, max_volume_x, count_sample);
int* x = new int[amount_x];
int* x_opt = new int[amount_x];
double f, f_opt;
int* mass_int_combi = new int[all_combination];
int counter_while, _j;
double _max_volume_x;
int count_while = 0;
int count_right = 0;
```

```
        while (count_while < 1) { //количество файлов //3
        double** mass = new double* [amount[count_while]]; //1000 //count_base
        for (int i = 0; i < amount[count_while]; i++) //count_base
        mass[i] = new double[all_combination]; //general
        for (int i = 0; i < all_combination; i++) //general
        mass_int_combi[i] = 0;
        counter_while = 0;
        while (counter_while < amount[count_while]) { //1000 //count_base
        std::random_device rd;
        std::mt19937 gen(rd());
        if (this->rB_unif_dist->Checked == true){
        std::uniform_real_distribution<>   rndC1(volume_x[0]   -   volume_x[0]   *   interes[5],
volume_x[0] + volume_x[0] * interes[5]);
        three_dimensional_mass[0] = rndC1(gen);
        std::uniform_real_distribution<>   rndC2(volume_x[1]   -   volume_x[1]   *   interes[6],
volume_x[1] + volume_x[1] * interes[6]);
        three_dimensional_mass[1] = rndC2(gen);
        std::uniform_real_distribution<>   rndC3(volume_x[2]   -   volume_x[2]   *   interes[7],
volume_x[2] + volume_x[2] * interes[7]);
        three_dimensional_mass[2] = rndC3(gen);
        std::uniform_real_distribution<>   rndC4(volume_x[3]   -   volume_x[3]   *   interes[8],
volume_x[3] + volume_x[3] * interes[8]);
        three_dimensional_mass[3] = rndC4(gen);
        std::uniform_real_distribution<>   rndC5(volume_x[4]   -   volume_x[4]   *   interes[9],
volume_x[4] + volume_x[4] * interes[9]);
        three_dimensional_mass[4] = rndC5(gen);


        std::uniform_real_distribution<> rndC11(x_cost[0] - x_cost[0] * interes[0], x_cost[0] +
x_cost[0] * interes[0]);
        three_dimensional_mass_1[0] = rndC11(gen);
        std::uniform_real_distribution<> rndC22(x_cost[1] - x_cost[1] * interes[1], x_cost[1] +
x_cost[1] * interes[1]);
        three_dimensional_mass_1[1] = rndC22(gen);
        std::uniform_real_distribution<> rndC33(x_cost[2] - x_cost[2] * interes[2], x_cost[2] +
x_cost[2] * interes[2]);
        three_dimensional_mass_1[2] = rndC33(gen);
        std::uniform_real_distribution<> rndC44(x_cost[3] - x_cost[3] * interes[3], x_cost[3] +
x_cost[3] * interes[3]);
        three_dimensional_mass_1[3] = rndC44(gen);
        std::uniform_real_distribution<> rndC55(x_cost[4] - x_cost[4] * interes[4], x_cost[4] +
x_cost[4] * interes[4]);
        three_dimensional_mass_1[4] = rndC55(gen);}
        else{
        std::normal_distribution<> rndC1(volume_x[0], (volume_x[0] * 0.1 * 2)* interes[5]);
```

```
three_dimensional_mass[0] = rndC1(gen);
std::normal_distribution<> rndC2(volume_x[1], (volume_x[1] * 0.1 * 2)* interes[6]);
three_dimensional_mass[1] = rndC2(gen);
std::normal_distribution<> rndC3(volume_x[2], (volume_x[2] * 0.1 * 2)* interes[7]);
three_dimensional_mass[2] = rndC3(gen);
std::normal_distribution<> rndC4(volume_x[3], (volume_x[3] * 0.1 * 2)* interes[8]);
three_dimensional_mass[3] = rndC4(gen);
std::normal_distribution<> rndC5(volume_x[4], (volume_x[4] * 0.1 * 2)* interes[9]);
three_dimensional_mass[4] = rndC5(gen);
std::normal_distribution<> rndC11(x_cost[0], (x_cost[0] * 0.1 * 2) * interes[0]);
three_dimensional_mass_1[0] = rndC11(gen);
std::normal_distribution<> rndC22(x_cost[1], (x_cost[1] * 0.1 * 2) * interes[1]);
three_dimensional_mass_1[1] = rndC22(gen);
std::normal_distribution<> rndC33(x_cost[2], (x_cost[2] * 0.1 * 2) * interes[2]);
three_dimensional_mass_1[2] = rndC33(gen);
std::normal_distribution<> rndC44(x_cost[3], (x_cost[3] * 0.1 * 2) * interes[3]);
three_dimensional_mass_1[3] = rndC44(gen);
std::normal_distribution<> rndC55(x_cost[4], (x_cost[4] * 0.1 * 2) * interes[4]);
three_dimensional_mass_1[4] = rndC55(gen);}
for (int i = 0; i < amount_x; i++)
x_opt[i] = x_min[i]; //0
for (int i = 0; i < amount_x; i++)
x[i] = x_min[i]; //0
f = 0;
f_opt = -1000;
_j = amount_x - 1;
_max_volume_x = 0;
while (true) {
if (x[_j] <= x_max[_j]) {
for (int i = 0; i < amount_x; i++)
_max_volume_x += three_dimensional_mass[i] * x[i];
if (_max_volume_x <= max_volume_x) {
for (int i = 0; i < amount_x; i++)
f += three_dimensional_mass_1[i] * x[i];
if (f_opt <= f) {
f_opt = f;
for (int i = 0; i < amount_x; i++)
x_opt[i] = x[i];
}
}
_j = amount_x - 1;
}
else {
x[_j] = x_min[_j]; //0
_j = _j - 1;
```

```
if (_j < 0)
break;
}
x[_j] += 1;
_max_volume_x = 0;
f = 0;
}
count_right = 0;
for (int i = 0; i < all_combination; i++) {
for (int k = 0; k < amount_x; k++)
if (count_sample[i][k] == x_opt[k])
count_right++;
if (count_right == amount_x) {
mass[mass_int_combi[i]][i] = f_opt;
mass_int_combi[i]++;
count_right = 0;
}
else count_right = 0;
}
counter_while++;
}
int max = 0;
for (int i = 0; i < all_combination; i++)
if (max < mass_int_combi[i])
max = mass_int_combi[i];
std::string* gyper = new std::string[1];
if (this->rB_unif_dist->Checked == true)
gyper[0] = path + "/" + std::to_string(amount[0]) + " uniform cost and volume.csv";
else gyper[0] = path + "/" + std::to_string(amount[0]) + " normal cost and volume.csv";
int count_plan = 0;
std::ofstream csvFileResult(gyper[count_while]);
for (int i = 0; i < all_combination; i++) {
if (mass_int_combi[i] != 0) {
count_plan++;
csvFileResult << "_";
for (int k = 0; k < amount_x; k++)
csvFileResult << count_sample[i][k];
if (i != all_combination - 1)
csvFileResult << "_,";
}
if (i == all_combination - 1)
csvFileResult << "_";
}
csvFileResult << "\n";
double* s_arif_sum = new double[all_combination];
```

```
for (int i = 0; i < all_combination; i++)
s_arif_sum[i] = 0;
for (int i = 0; i < max; i++) {
for (int mk = 0; mk < all_combination; mk++) {
if (mass_int_combi[mk] != 0) {
if (mass_int_combi[mk] > i) {
s_arif_sum[mk] += mass[i][mk];
csvFileResult << mass[i][mk];
}
if (mk != all_combination - 1)
csvFileResult << ",";
}
if (mk == all_combination - 1)
csvFileResult << "\n";
}
}
for (int i = 0; i < all_combination; i++) {
if (mass_int_combi[i] != 0) {
csvFileResult << "_";
for (int k = 0; k < amount_x; k++)
csvFileResult << count_sample[i][k];
if (i != all_combination - 1)
csvFileResult << "_,";
}
if (i == all_combination - 1)
csvFileResult << "_";
}
csvFileResult << "\n";
for (int mk = 0; mk < all_combination; mk++) {
if (mass_int_combi[mk] != 0) {
csvFileResult << Convert::ToDouble(mass_int_combi[mk]) / Convert::ToDouble(amount[0]) *
100;
if (mk != all_combination - 1)
csvFileResult << ",";
}
if (mk == all_combination - 1)
csvFileResult << ",Кількість з'явлень\n";
}
for (int mk = 0; mk < all_combination; mk++) {
if (mass_int_combi[mk] != 0) {
csvFileResult << (s_arif_sum[mk] / Convert::ToDouble(mass_int_combi[mk]));
if (mk != all_combination - 1)
csvFileResult << ",";
}
if (mk == all_combination - 1)
```

```
csvFileResult << ",Середнєариф\n";
}
double counters_;
for (int mk = 0; mk < all_combination; mk++) {
if (mass_int_combi[mk] != 0) {
counters_ = 0;
for (int i = 0; i < max; i++)
if (mass_int_combi[mk] > i)
if (mass[i][mk] > 30000)
counters_ += 1;
csvFileResult << (counters_ / Convert::ToDouble(mass_int_combi[mk])) * 100;
if (mk != all_combination - 1)
csvFileResult << ",";
}
if (mk == all_combination - 1)
csvFileResult << ",Імовірність >30000\n";
}
for (int mk = 0; mk < all_combination; mk++) {
if (mass_int_combi[mk] != 0) {
counters_ = 0;
for (int i = 0; i < max; i++)
if (mass_int_combi[mk] > i)
if (mass[i][mk] > 35000)
counters_ += 1;
csvFileResult << (counters_ / Convert::ToDouble(mass_int_combi[mk])) * 100;
if (mk != all_combination - 1)
csvFileResult << ",";
}
if (mk == all_combination - 1)
csvFileResult << ",Імовірність >35000\n";
}
csvFileResult.close();
count_while++;
}
}
}
}
else
MessageBox::Show("Error! \nSelect file path");
this->progressBar3->Value = 100;};

private: int Amount_Combo(int* a_maksimalno, int* a_minimalno, double* prototip_a, int
kilkist_a, double summa_prototip_a) {
int* massiv_a = new int[kilkist_a];
for (int i = 0; i < kilkist_a; i++)
```

```
        massiv_a[i] = a_minimalno[i];
        int zagalna = 0, vihid_z_tsiklu = 0;
        double zagalna_summa_combi;
        bool true_if = false;
        while (true) {
        zagalna_summa_combi = 0;
        if (true_if == true)
        massiv_a[kilkist_a - 1] += 1;
        else true_if = true;
        for (int i = 0; i < kilkist_a; i++)
        zagalna_summa_combi += massiv_a[i] * prototip_a[i];
        if (zagalna_summa_combi <= summa_prototip_a)
        zagalna++;

        for (int i = 1; i < kilkist_a; i++)
        if (massiv_a[i] > a_maksimalno[i]) {
        massiv_a[i - 1] += 1;
        massiv_a[i] = a_minimalno[i]; //0
        true_if = false;
        }
        for (int i = 0; i < kilkist_a; i++)
        if (massiv_a[i] == a_maksimalno[i])
        vihid_z_tsiklu++;
        if (vihid_z_tsiklu == kilkist_a)
        break;
        else vihid_z_tsiklu = 0;}
        return zagalna;}

        private: void All_Combo(int* a_maksimalno, int* a_minimalno, double* prototip_a, int
kilkist_a, double summa_prototip_a, int** massiv_z_combi) {
        int* massiv_a = new int[kilkist_a];
        for (int i = 0; i < kilkist_a; i++) {
        massiv_a[i] = a_minimalno[i]; //0
        }
        int zagalna = 0, vihid_z_tsiklu = 0;
        double zagalna_summa_combi;
        bool true_if = false;
        while (true) {
        zagalna_summa_combi = 0;
        if (true_if == true)
        massiv_a[kilkist_a - 1] += 1;
        else true_if = true;
        for (int i = 0; i < kilkist_a; i++)
        zagalna_summa_combi += massiv_a[i] * prototip_a[i];
        if (zagalna_summa_combi <= summa_prototip_a) {
```

```
for (int i = 0; i < kilkist_a; i++)
massiv_z_combi[zagalna][i] = massiv_a[i];
zagalna++;
}
for (int i = 1; i < kilkist_a; i++)
if (massiv_a[i] > a_maksimalno[i]) {
massiv_a[i - 1] += 1;
massiv_a[i] = a_minimalno[i]; //0
true_if = false;
}
for (int i = 0; i < kilkist_a; i++)
if (massiv_a[i] == a_maksimalno[i])
vihid_z_tsiklu++;
if (vihid_z_tsiklu == kilkist_a)
break;
else
vihid_z_tsiklu = 0;}}
}
```