НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки

Кафедра інформаційних систем та технологій

Індивідуальний дослідницький проєкт

на здобуття ступеня бакалавра
за освітньо-професійною програмою «Інформаційні управляючі системи та технології»
спеціальності 126 «Інформаційні системи та технології»

на тему: «Інформаційна система з підтримки дослідження задачі складання розкладу виконання робіт паралельними машинами, для яких виконується умова вкладеності»

Виконав: студент *IV* курсу, групи *ІС-82*

Шенгелія Володимир Олександрович _____
(прізвище, ім'я, по батькові) (підпис)

Керівник *доц., к.т.н., доц. Жданова Олена Григорівна* _____
(посада, науковий ступінь, вчене звання, прізвище, ім'я, по батькові) (підпис)

Засвідчую, що у цьому проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2022 року

Національний технічний університет України

“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет інформатики та обчислювальної техніки

Кафедра інформаційних систем та технологій

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 126 «Інформаційні системи та технології»

Освітньо-професійна програма «Інформаційні управляючі системи та технології»

З А В Д А Н Н Я
на індивідуальний дослідницький проєкт студенту

Шенгелія Володимиру Олександровичу
(прізвище, ім’я, по батькові)

1. Тема проєкту *«Інформаційна система з підтримки дослідження задачі складання розкладу виконання робіт паралельними машинами, для яких виконується умова вкладеності»*
керівник проєкту *Жданова Олена Григорівна, к.т.н., доцент*
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

2. Термін подання студентом проєкту “*15*” *червня* 2022 року

3. Вихідні дані до проєкту

Технічне завдання

4. Зміст пояснювальної записки

1. Загальні положення: основні визначення та терміни, опис предметного середовища, огляд ринку програмних продуктів, постановка задачі

2. Інформаційне забезпечення: вхідні дані, вихідні дані, опис структури бази даних

3. Математичне забезпечення: математична постановки задачі, обґрунтування та опис методу розв’язання, методологія дослідження задачі

4. Програмне та технічне забезпечення: засоби розробки, вимоги до технічного забезпечення, архітектура програмного забезпечення, побудова звітів

5. Технологічний розділ: керівництво користувача, методика випробувань програмного продукту

5. Перелік графічного матеріалу

1. Схема структурна варіантів використання

2. Схема структурна діяльності «Планування розкладу базуючись на даних з файлу»

3. Схема структурна діяльності «Планування розкладу базуючись на випадково згенерованих даних»

4. Схема структурна діяльності «Розрахунок значень цільових функцій при розв'язанні задач різної розмірності»

5. Схема структурна діяльності «Підрахунок середнього часу витраченого на розв'язання задач різної розмірності»

6. Схема бази даних

6. Дата видачі завдання *«1» грудня 2021 року*

Календарний план

№ з/п	Назва етапів виконання проєкту	Термін виконання етапів проєкту	Примітка
1.	*Вивчення рекомендованої літератури*	*03.05.2022*	
2.	*Аналіз існуючих методів розв'язання задачі*	*06.05.2022*	
3.	*Постановка та формалізація задачі*	*10.05.2022*	
4.	*Розробка інформаційного забезпечення*	*13.05.2022*	
5.	*Алгоритмізація задачі*	*17.05.2022*	
6.	*Обґрунтування використовуваних технічних засобів*	*20.05.2022*	
7.	*Розробка програмного забезпечення*	*26.05.2022*	
8.	*Налагодження програми*	*02.06.2022*	
9.	*Виконання графічних документів*	*06.06.2022*	
10.	*Оформлення пояснювальної записки*	*10.06.2022*	
11.	*Подання індивідуального дослідницького проєкту*	*15.06.2022*	

Студент Володимир ШЕНГЕЛІЯ

Керівник Олена ЖДАНОВА

Пояснювальна записка
до індивідуального дослідницького проєкту на тему: «Інформаційна система з підтримки дослідження задачі складання розкладу виконання робіт паралельними машинами, для яких виконується умова вкладеності»

Київ – 2022 року

АНОТАЦІЯ

Структура та обсяг роботи. Пояснювальна записка індивідуального дослідницького проєкту складається з п’яти розділів, містить 36 рисунків, 5 таблиць, 18 джерел, 1 додаток.

Індивідуальний дослідницький проєкт присвячено реалізації інформаційної системи з підтримки дослідження задачі складання розкладу виконання робіт паралельними машинами, для яких виконується умова вкладеності.

Метою розробки інформаційної системи є дослідження алгоритмів складання розкладу виконання робіт, які використовуються для розв’язання поставленої задачі, порівняння їх ефективності та швидкодії.

Розділ інформаційного забезпечення описує вхідні та вихідні дані, а також структуру бази даних у реалізованому програмному продукті.

Розділ математичного забезпечення надає опис математичної моделі задачі оптимізації, алгоритмів її розв’язання, плану та результатів проведення експериментів, які визначають найбільш ефективний алгоритм для подальшого практичного використання.

Розділ програмного забезпечення містить опис розробленого веб-застосування та обґрунтування вибору засобів для його розробки.

Технологічний розділ описує керівництво користувача та результати тестів, проведених під час реалізації веб-застосування.

СКЛАДАННЯ РОЗКЛАДІВ, ПАРАЛЕЛЬНІ МАШИНИ, УМОВА ВКЛАДЕНОСТІ, ПРАВИЛО НАЙМЕНШ ГНУЧКОЇ РОБОТИ, МІНІМІЗАЦІЯ ЧАСУ ЗАВЕРШЕННЯ ВИКОНАННЯ РОБІТ.

ABSTRACT

Structure and scope of work. The explanatory statement of the diploma project consists of five sections, contains 36 pictures, 5 tables, 18 references, 1 appendix.

The individual research project is devoted to the implementation of the information system to support the research of the problem of scheduling jobs with parallel machines, for which the condition of nesting is met.

The objective of information system development is to research the algorithms for jobs scheduling, which are used to solve the problem, comparing their efficiency and speed.

The information management section describes the structure of input and output data, the structure of the database in the implemented software product.

The section of mathematical management provides a description of the mathematical model of the optimization problem, algorithms for its solution, plan and results of experiments, which determine the most efficient algorithm for further practical use.

The software management section contains a description of the developed web application and justification for the choice of tools for its development.

The technology management section provides user manuals and test results performed during the implementation of the web application.

SCHEDULING, PARALLEL MACHINES, CONDITION OF NESTING, RULE OF LEAST FLEXIBLE JOB, MINIMIZATION OF TIME FOR JOBS EXECUTION.

ЗМІСТ

ВСТУП

Індивідуальний дослідницький проєкт спрямований на розробку інформаційної системи з підтримки дослідження задачі оптимізації, а саме складання розкладу виконання робіт паралельними машинами, для яких виконується умова вкладеності.

Умова вкладеності означає, що дана робота може бути виконана лише на множині машин заданій для неї.

Для будь-яких двох робіт, що мають умову вкладеності, у множин машин виконується тільки одна і лише одна із наступних умов: множини є рівноцінними; одна із множин включає іншу; при перетині множин машин знаходиться лише порожня множина.

За цільову функцію, яку слід мінімізувати, було обрано критерій, що є найбільш логічним у сфері управління виробництвом – це момент завершення виконання останньої роботи.

Сенс практичного застосування полягає у розв’язанні оптимізаційної задачі задля надання можливості планувати ефективне завантаження наявних трудових або апаратних ресурсів на виробництві.

Розроблені алгоритми під час розв’язання поставленої задачі можуть бути використані в якості компонентів систем, що є націленими на оперативне й оперативно-календарне планування виробництва.

Для наведення практичного прикладу використання напрацювань у розв’язанні поставленої задачі візьмемо за приклад будь-яке стратегічне підприємство критичного значення, де затримки у часі, який затрачено на виготовлення виробів, можуть коштувати іноді навіть життя.

Саме дослідження та розв’язання задач із розділу теорії складання розкладів допомагає подолати зазначенні вище проблем і також банально оптимізувати працю цілого виробництва.

Практичне значення одержаних результатів. Розроблено два евристичні алгоритми, проведено дослідження їх точності та швидкодії. Реалізовано веб-застосування для інформаційної системи з підтримки дослідження задачі складання розкладу виконання робіт паралельними машинами, для яких виконується умова вкладеності.

1 ЗАГАЛЬНІ ПОЛОЖЕННЯ

1.1 Опис інформаційної системи

Постала необхідність у створенні інформаційної системи, яка присвячена дослідженню оптимізаційної задачі з теорії розкладів; постановку задачі, яка є метою дослідження, наведено у розділі 1.2.

До функціоналу інформаційної системи було поставлено наступні вимоги:

– розв’язання задачі за допомогою алгоритмів, які були розроблені; візуалізація результатів виконання робіт машинами на екрані у формі діаграми Ґанта;
– генерація довільної задачі, якій задано розмірність, щоб у подальшому отримати її розв’язок алгоритмами, які були розроблені;
– проведення експериментів для великої кількості задач, які мають різну розмірність.

За мету у проведені експериментів поставлено:

– проведення дослідження для встановлення точності алгоритмів;
– проведення дослідження для встановлення швидкодії алгоритмів.

1.2 Постановка задачі теорії розкладів, що досліджується

Нехай маємо m машин, що працюють паралельно та призначені для виконання робіт (позначаються M_i , $i = \overline{1,m}$).

Є множина з n робіт, кожна робота із цієї множини робіт характеризується тривалістю виконання p_j, для будь-якої з робіт відсутній директивний термін її виконання.

Для роботи j ($j = \overline{1,n}$) задана множина машин M_j , на яких ця робота може бути виконана. При цьому припускається, що ці множини є *вкладеними*,

тобто для довільних робіт j та k виконується одна і тільки одна із наступних умов:

$$M_j = M_k ;$$
$$M_j \subset M_k ;$$
$$M_k \subset M_j ;$$
$$M_j \cap M_k = \emptyset.$$

Тривалість виконання будь-якої роботи дорівнює 1 або 2.

Припускається, що поставлена задача є задачею без переривань.

Припускається, що дані машини мають однакову продуктивність, через це коефіцієнт продуктивності відсутній.

За прийнятний час виконання алгоритму знайти розклад, в якому час завершення виконання останньої роботи (загальний час виконання усіх робіт) буде мінімальним чи наближеним до мінімального.

1.2.1 Приклад індивідуальної задачі

Компанія «Укроборонпром» отримала термінове замовлення на виготовлення деяких 12 виробів, час роботи над кожним з яких складає 1 або 2 доби. Виготовлення кожного виробу можливе тільки на певній множині верстатів. Ці 12 робіт повинні бути розподілені між 3 верстатами.

Припускається, що дедлайни для виготовлення цих виробів не встановлені, а верстати не мають обмеження на кількість створених виробів.

Знайти розклад виготовлення виробів, за якого час завершення виготовлення останнього виробу (загальний час виготовлення всіх виробів) буде мінімальним.

Тривалості виготовлення виробів та множини верстатів для виробів подані в таблиці 1.1.

Таблиця 1.1 – Тривалості виготовлення виробів та множини верстатів для виробів із прикладу

Порядковий номер виробу	Тривалість виготовлення	Множина верстатів
1	1	1
2	1	3
3	1	2, 3
4	2	1, 2, 3
5	1	2
6	1	1
7	2	2, 3
8	2	2
9	1	3
10	2	1, 2, 3
11	1	2
12	2	1, 2, 3

На рисунках 1.1, 1.2 наведено можливі розклади виготовлення виробів (розв’язки) із індивідуальної задачі.

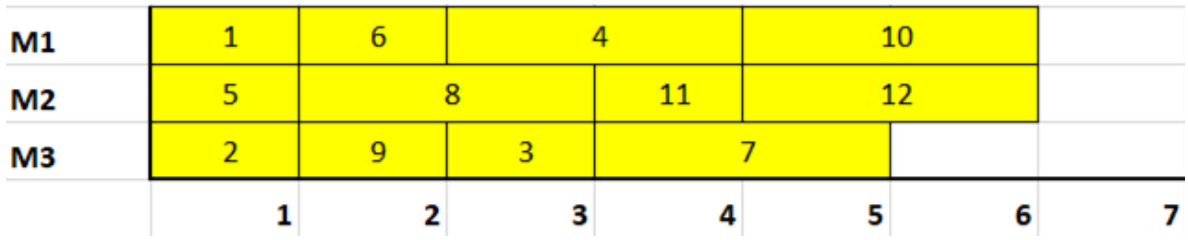

Рисунок 1.1 – Перший допустимий розклад виготовлення виробів

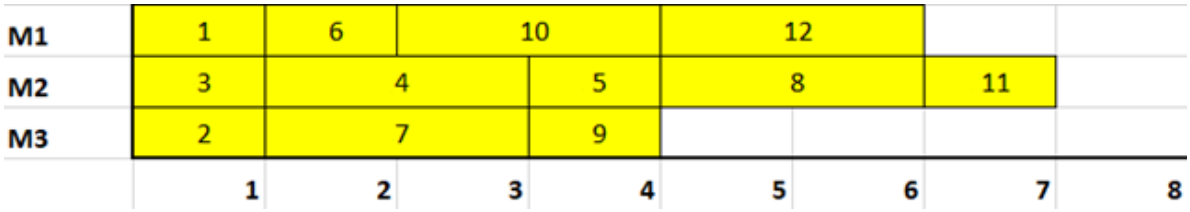

Рисунок 1.2 – Другий допустимий розклад виготовлення виробів

У першому випадку загальний термін виготовлення виробів дорівнює 6 діб, у другому – 7 діб. Сенс оптимізації даної задачі – мінімізація часу, за який буде завершена остання робота.

1.3 Призначення та мета розробки

1.3.1 Призначення розробки

Призначенням розробленої інформаційної системи є підтримка дослідження оптимізаційної задачі із теорії розкладів, а саме задачі складання розкладів, коли роботи виконуються паралельними машинами, для яких наявна умова вкладеності.

1.3.2 Мета та задачі розробки

За мету в розробці цієї інформаційної системи було поставлено підтримку процесів, що направлені на дослідження й порівняння за ефективністю алгоритмів, що використовуються для розв’язання даної задачі.

Реалізація наступних задач є необхідною для досягнення сформованої вище мети:

- зчитування вихідної задачі з файлу;
- генерація довільної задачі заданої розмірності із можливістю подальшого її розв’язання обраними алгоритмами;
- складання розкладів виконання робіт машинами за умови, що роботи виконуються на вкладених множинах машин, а машини є паралельними;
- збереження результатів складання індивідуальних задач заданої розмірності для подальшого порівняння значень отриманих цільових функцій та часу, необхідного для розв’язання поставленої задачі;
- створення можливості запуску алгоритмів для заданої кількості прогонів, щоб надати можливість отримати звіт роботи алгоритмів, на базі якого можна визначити найбільш ефективний з них.

Розроблені алгоритми можуть бути використані в якості компонентів систем, що є націленими на оперативне й оперативно-календарне планування виробництва.

1.4 Опис функціональної моделі

Створений програмний продукт складається із наступних функціональних елементів для дослідження індивідуальної задачі:

- введення за допомогою текстового файлу даних для індивідуальної задачі; генерація даних для індивідуальних задач;
- обчислення розробленими алгоритмами значень цільових функцій задач, для яких було задано діапазон розмірностей;
- визначення для заданого діапазону розмірностей середнього часу розв’язання задач та оптимального значення цільової функції.

У графічних матеріалах наведено діаграму варіантів використання, відому як use case diagram, а також схему бази даних до розробленого програмного продукту.

1.5 Огляд аналогів

Задача, яка була обрана для дослідження, є абсолютно новою й унікальною через наявні у неї певні додаткові обмеження (довжина робіт), а також узагальнення (вкладеність машин). Не існує методів, які б уже були створені, для розв’язання саме даної задачі із розділу складання розкладів.

Із цього слідує відсутність аналогів інформаційної системи, що розв’язує поставлену задачу, та інформаційної системи, яка спрямована на підтримку дослідження поставленої задачі.

Висновок до розділу

У цьому розділі описано сформульовану змістовну постановку задачі, дослідження й реалізацію розв’язання якої представлено у вигляді програмного продукту. Наведено приклад практичної індивідуальної задачі та певних допустимих її розв’язків. Задача є новою, тому аналогів програмних продуктів до створеного не існує.

2 ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ

2.1 Вхідні дані

Вхідні дані можуть бути передані двома способами: через текстовий файл або згенеровані випадковим чином.

У випадку зчитування даних із текстового файлу, кожні три рядки файлу визначають наступні характеристики, якими описується робота:

– перший рядок відповідає за номер роботи по порядку;
– другий рядок відповідає за тривалість виконання роботи;
– у третьому рядку зазначено множину номерів машин, де можливе виконання цієї роботи.

При передачі вхідних даних за допомогою файлу у програмі є валідація кількості рядків вхідного файлу для коректного зчитування даних. Самі дані файлу також перевіряються на відповідність до вимог опису робіт.

У випадку генерації вхідних даних випадковим чином, користувач вводить до веб-форми кількість робіт і машин, відсоток робіт із довжиною 2 та обирає тип генерації множин машин. Після цього програма визначає властивості робіт таким чином, щоб вони задовольняли обмеження, описані у розділі 1.

Тестовий приклад текстового файлу наведено на рисунку 2.1. Дані у цьому файлі описують індивідуальну задачу на вхід до якої подано 16 робіт і 4 машини, на яких вони можуть бути виконані. У таблиці 2.1 можна знайти опис робіт: їх порядковий номер, тривалість виконання роботи та відповідно самі множини машин, на яких виконується ця роботи.

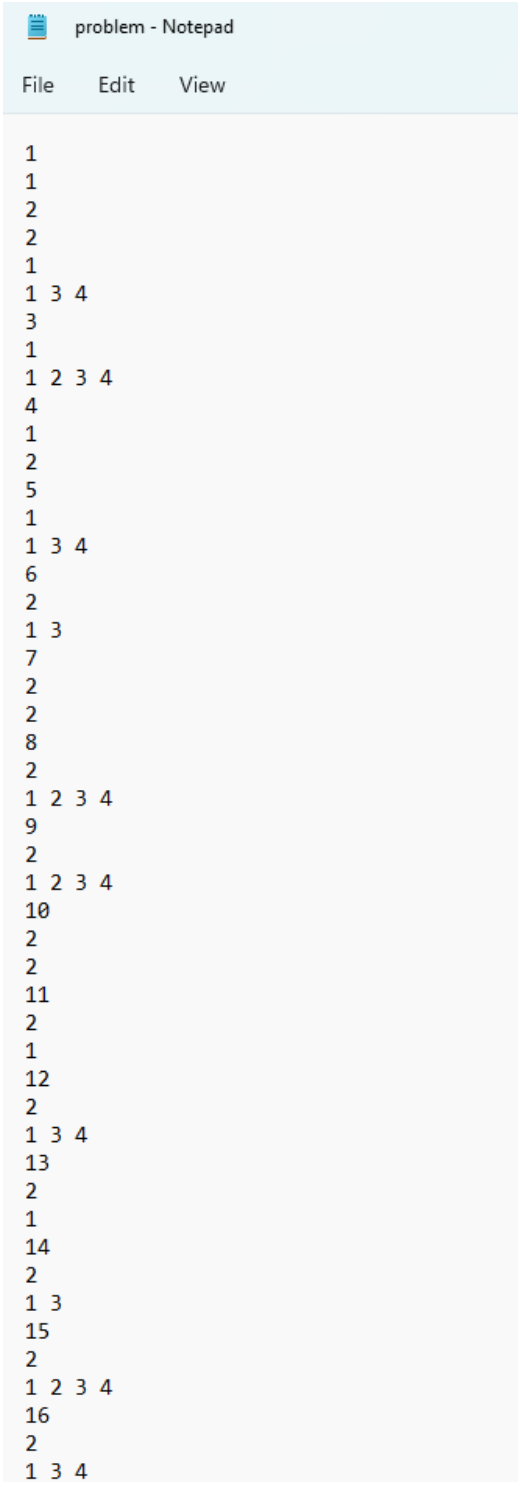

Рисунок 2.1 – Приклад тестового текстового файлу

Таблиця 2.1 – Тривалості виготовлення виробів та множини верстатів для виробів із текстового файлу

Порядковий номер виробу	Тривалість виготовлення	Множина верстатів
1	1	2
2	1	1, 3, 4
3	1	1, 2, 3, 4
4	1	2
5	1	1, 3, 4
6	2	1, 3
7	2	2
8	2	1, 2, 3, 4
9	2	1, 2, 3, 4
10	2	2
11	2	1
12	2	1, 3, 4
13	2	1
14	2	1, 3
15	2	1, 2, 3, 4
16	2	1, 3, 4

2.2 Вихідні дані

Можливі два варіанти використання програмного продукту «Розв’язання задачі за даними на вхід, що згенеровані випадково» та «Розв’язання задачі за даними на вхід, що зберігаються у файлі».

Вибір опції «Розв’язання задачі за даними на вхід, що згенеровані випадково» призводить до того, що на екрані користувача виводяться наступні дані:

– розв’язок індивідуальної задачі за допомогою обраних із розроблених алгоритмів у вигляді таблиці (діаграми Ґанта);
– перегляд вхідних даних задачі, що також надає можливість збереження вхідних даних у файл.

Вибір опції «Розв’язання задачі за даними на вхід, що зберігаються у файлі» призводить до того, що на екрані користувача виводяться наступні дані:

– повідомлення про успішність перевірки даних у вхідному файлі щодо відповідності структурі запису робіт;
– розв’язок індивідуальної задачі за допомогою обраних із розроблених алгоритмів у вигляді таблиці (діаграми Ґанта);
– перегляд вхідних даних задачі, що також надає можливість повторного збереження вхідних даних у файл.

Результати експериментів, що були проведені, так само як і при виборі опції «Розв’язання задачі за даними на вхід, що згенеровані випадково» надають наступні результати:

– розв’язок індивідуальної задачі за допомогою обраних із розроблених алгоритмів у вигляді таблиці (діаграми Ґанта);
– перегляд вхідних даних задачі, що також надає можливість збереження вхідних даних у файл.

2.3 Структура масивів інформації

У метод, що розв’язує індивідуальну задачу, подається об’єкт Problem (задача), він складається із двох масивів (однозв’язних списків):

– масив робіт, до складу якого входять елементи, що відносяться до типу Task (робота), які зберігають інформацію про машини, на яких може виконуватись дана робота, та час необхідний для її виконання;
– масив машин, до складу якого входять елементи, що відносяться до типу Machine (машина), які зберігають інформацію про роботи, що вже виконує дана машина, та загальний час її роботи на даний момент.

2.4 Опис структури бази даних

База даних призначена для зберігання результатів роботи алгоритмів для подальшого відображення їх із метою надання необхідної інформації за кожним із розроблених алгоритмів під час їх дослідження.

Структура бази даних наведена на рисунку 2.2, в ній присутня лише одна таблиця, що зберігає усю необхідну інформацію у зручному для представлення вигляді [5].

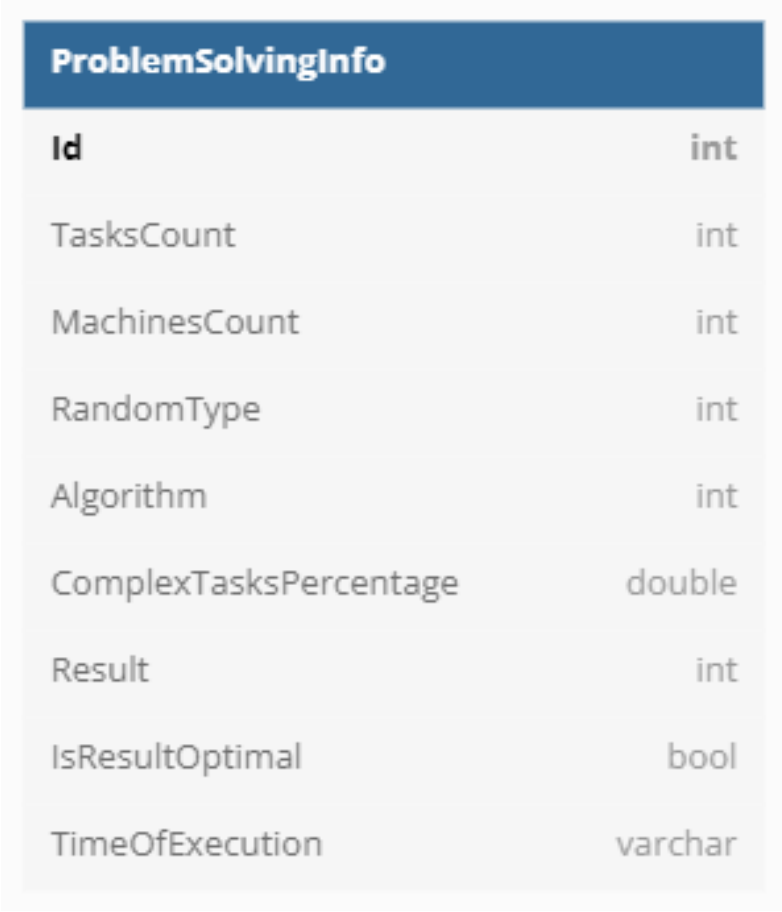

Рисунок 2.2 – Структура бази даних

Наведення опису даних із таблиці ProblemSolvingInfo:

– Id – цілочисельне значення, що відповідає за збереження індексу запису про розв’язання конкретної задачі;

– TasksCount – цілочисельне значення, що відповідає за збереження кількості робіт, що надані для планування у даній індивідуальній задачі;
– MachinesCount – цілочисельне значення, що відповідає за збереження кількості машин, що надані для планування у даній індивідуальній задачі;
– RandomType – цілочисельне значення, що відповідає за збереження типу генерації наборів вкладених машин для кожної із робіт, а саме за кількість і розподіл вкладених машин, де 0 – пірамідальний розподіл, 1 – розподіл випадковим чином;
– Algorithm – цілочисельне значення, що відповідає за збереження інформації про обраний для розв’язку індивідуальної задачі алгоритм, де 0 – оптимальний алгоритм, 1 – алгоритм A, 2 – алгоритм B;
– ComplexTasksPercentage – раціональне значення, що відповідає за збереження у відсотках відносної кількості робіт із часом виконання, що дорівнює 2;
– Result – цілочисельне значення, що відповідає за збереження отриманого розв’язку індивідуальної задачі обраним із розроблених алгоритмів;
– IsResultOptimal – булеве значення, що відповідає за збереження інформації щодо оптимальності отриманого розв’язку, де 0 – розв’язок не є оптимальним, 1 – розв’язок є оптимальним;
– TimeOfExecution – рядкове значення, що відповідає за збереження часу, що було витрачено на розв’язання даної індивідуальної задачі обраним алгоритмом, у мс.

Висновок до розділу

Цей розділ присвячено опису вхідних та вихідних даних, структури бази даних та їх використанню у процесі створення програмного продукту.

У цьому розділі було детально описано формат даних у вхідному файлі, який може бути використаний програмним продуктом у якості вхідних даних для розв’язання індивідуальної задачі, надано практичний приклад такого файлу.

Формати виведення описані для вихідних даних, а також для варіантів використання, які пов’язані з розв’язання як окремих задач, так із дослідженням розроблених алгоритмів, що представляє собою багатократне розв’язання задач, що мають певну задану розмірність.

Надано структуру бази даних для збереження результатів попередніх прогонів алгоритмів на заданих вхідних даних, із яких генерувалися індивідуальні задачі.

3 МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ

3.1 Математична постановка задачі

Дано:

m – кількість машин, що працюють паралельно

n – кількість робіт

p_i – тривалість виконання робіт, $p_i \in \{1,2\}$, де $i = \overline{1,n}$

M_i – задана множина машин для виконання i роботи, де $i = \overline{1,n}$

Множини є вкладеними, тобто для довільних робіт i та k виконується одна і тільки одна із наступних умов:

$$M_i = M_k ;$$
$$M_i \subset M_k ;$$
$$M_k \subset M_i ;$$
$$M_i \cap M_k = \emptyset.$$

Змінні:

J_l – множина робіт, яка виконується на l машині, де $l = \overline{1,m}$

Цільова функція:

Знайти мінімальний час завершення виконання всіх робіт:

$$max\{\sum_{i\in J_1} p_i; ..; \sum_{i\in J_m} p_i\} \to min$$

Обмеження:

Згідно з умовою задачі множини машин для виконання певної роботи є вкладеними.

Множини робіт, які виконуються на машинах не перетинаються:

$$\forall r,s \ J_r \cap J_s = \emptyset, \text{де } r,s = \overline{1,m}$$

Об’єднання множин робіт, які виконуються на $\overline{1,m}$ машинах, складає множину всіх робіт:

$$\bigcup_{l=1}^{m} J_l = \{1,2,..,n\}$$

3.2 Обґрунтування методу розв’язання

Задачі $Pm \mid p_j = 1,2\ Mj \mid Cmax$ (у яких машини є ідентичними та паралельними, роботи мають час виконання один або два, а цільова функція відповідає часу завершення останньої роботи) є NP-повними [4] у загальному випадку.

Частковий випадок даної задачі, $Pm \mid p_j = 1\ Mj \mid Cmax$ (коли час виконання будь-якої з робіт дорівнює 1), є поліноміальною задачею, котра розв’язується за допомогою правила найменш гнучкої роботи (*least flexible job first – LFJF*). У такому випадку правило гнучкої роботи є повним, це значить те, що гарантовано буде отримано оптимальний розв’язок. За напрацюваннями Пінедо [1], правило гнучкої роботи для випадку $Pm \mid p_j = 1\ Mj \mid Cmax$ є оптимальним за умови, коли множини M_j є вкладеними.

Загальна ідея правила найменш гнучкої роботи полягає в тому, що кожного разу, коли машина звільнюється, серед наявних робіт обирається та, яку можна обробити на найменшій кількості машин, тобто найменш гнучка робота. Це правило є досить грубим, оскільки не вказує, наприклад, яку машину потрібно обрати першою, коли кілька машин можуть бути обрані одночасно.

Правило найменш гнучкої роботи було обрано, бо воно є P-складним та оптимальним для даного часткового випадку $Pm \mid p_j = 1\ Mj \mid Cmax$. Із цього слідує, що можливо розробити алгоритм, який базується на правилі найменш

гнучкої роботи та дає можливість отримати розв'язок, який буде наближеним для узагальненого випадку постановки задачі $Pm \mid p_j = 1{,}2\ Mj \mid Cmax$ [1].

3.3 Розроблені алгоритми розв'язання задачі

Було розроблено два алгоритми, які базуються на правилі найменш гнучкої роботи та дають можливість отримати розв'язок, що буде наближеним до оптимального для узагальненого випадку даної задачі.

3.3.1 Опис алгоритму А

3.3.1.1 Розробка алгоритму розв'язання задачі

Ідея алгоритму полягає у тому, що на кожному кроці, розглядаючи множину робіт, обирається робота із найменшою кількістю машин, на яких вона може виконуватись. Обрана робота призначається на машину із найменшим завантаженням серед тих, на яких вона може виконуватись. Повністю співпадає із ідеєю поліноміальної задачі, яка вирішується за правилом найменш гнучкої роботи.

Даний алгоритм належить до класу жадібних.

Схема алгоритму А наведена на схемі 3.1.

Схема 3.1 – Схема розробленого алгоритму А

Вхід: n – кількість робіт;
m – кількість машин;
$p_1, p_2, \dots, p_n$ - тривалість виконання;
$M_1, M_2, \dots, M_n$ – множини машин для виконання робіт.
Вихід: $J_1, J_2, \dots, J_m$ – множини робіт, які виконуються на машинах;
C_{max} – час закінчення виконання всіх робіт.
Упорядкувати роботи так, що $|M_1| \le |M_2| \le .. \le |M_n|$
for i:=1 **to** m **do**
$J_i := \emptyset$
end for

Продовження схеми 3.1

```
11 for i:=1 to n do
12        //Призначити виконання роботи найменш завантаженій машині
13        J_leastLoad(Mi) := J_leastLoad(Mi) ∪ i
14 end for
15 C_max := maxTime()
16 //Пошук найменш завантаженої машини
17 function leastLoad(M)
18        minTime := ∞
19        leastLoadIndex := ∅
20 for всі індекси машин a ∈ M do
21        currentTime := 0
22        for всі індекси робіт b ∈ J_a do
23              currentTime := currentTime + p_b
24        end for
25        if (currentTime < minTime) then {
26              minTime := currentTime
27              leastLoadIndex = a
28        }
29 end for
30 return leastLoadIndex
31 end function
32 //Визначення часу закінчення всіх робіт
33 function maxTime()
34 time :=0
35 for i:=1 to m do
36        currentTime := 0
37        for всі індекси робіт a ∈ J_i do
38              currentTime := currentTime + p_a
39        end for
40        if (currentTime > time) then
41              time := currentTime
42 end for
```

Продовження схеми 3.1

```
43 return time
44 end function
```

3.3.1.2 Оцінка трудомісткості алгоритму

На першому кроці розв'язання відбувається сортування даних, його часова складність становить $O(nlog_n)$.

На другому кроці відбувається ініціалізація даних у циклі, часова складність становить $O(n)$.

На наступному кроці в циклі для кожної роботи визначається машина, на якій ця робота повинна виконуватись, часова складність становить $O(n)$. У цьому циклі присутній виклик функції, що також має цикл, тому загальна часова складність становить $O(n^2)$.

На фінальному кроці відбувається визначення часу, за який виконуються всі роботи. Для кожної машини у циклі підраховується тривалість її роботи, потім обирається найбільший час виконання роботи. Загальна складність цього кроку становить $O(n^2)$, бо для підрахунків використовується цикл із вкладеним циклом.

Відповідно, асимптотична оцінка трудомісткості алгоритму за критерієм часу становить $O(n^2)$ [3].

3.3.2 Опис алгоритму B

3.3.2.1 Розробка алгоритму розв'язання задачі

Ідея алгоритму полягає у тому, що на кожному кроці, коли розглядається множина робіт, то обирається робота, яка має найменшу кількість машин, де вона може виконуватись, та найбільший час виконання роботи. Робота, яку обрали, призначається на машину, що має найменше завантаженням серед тих,

де ця робота може виконуватись. Даний алгоритм є модифікацією алгоритму А, який було засновано на правилі найменш гнучкої роботи.

Алгоритм В належить також до класу жадібних.

Схему алгоритму В наведено на схемі 3.2.

Вхід: n – кількість робіт;
m – кількість машин;
$p_1, p_2, \dots, p_n$ - тривалість виконання;
$M_1, M_2, \dots, M_n$ – множини машин для виконання робіт.
Вихід: $J_1, J_2, \dots, J_m$ – множини робіт, які виконуються на машинах;
C_{max} – час закінчення виконання всіх робіт.
Упорядкувати роботи так, щоб одночасно виконувались дві умови:
$|M_1| \leq |M_2| \leq .. \leq |M_n|$ та $p_1 \geq p_2 \geq .. \geq p_n$.

```
for i:=1 to m do
        J_i := ∅
end for
for i:=1 to n do
        //Призначити виконання роботи найменш завантаженій машині
        J_leastLoad(Mi) := J_leastLoad(Mi) ∪ i
end for
C_max := maxTime()
//Пошук найменш завантаженої машини
function leastLoad(M)
        minTime := ∞
        leastLoadIndex := ∅
for всі індекси машин a ∈ M do
        currentTime := 0
        for всі індекси робіт b ∈ J_a do
                currentTime := currentTime + p_b
        end for
        if (currentTime < minTime) then {
                minTime := currentTime
```

Продовження схеми 3.2

```
27                  leastLoadIndex = a
28          }
29 end for
30 return leastLoadIndex
31 end function
32 //Визначення часу закінчення всіх робіт
33 function maxTime()
34 time :=0
35 for i:=1 to m do
36          currentTime := 0
37          for всі індекси робіт a ∈ J_i do
38                  currentTime := currentTime + p_a
39          end for
40          if (currentTime > time) then
41                  time := currentTime
42 end for
43 return time
44 end function
```

3.3.2.1 Оцінка трудомісткості алгоритму

На першому кроці розв’язання дані сортуються, часова складність становить $O(nlog_n)$.

Під час другого кроку дані у циклі ініціалізуються, часова складність становить $O(n)$.

На третьому кроці для кожної роботи в циклі визначається машина, де ця робота буде виконуватись, часова складність становить $O(n)$. У даному циклі є виклик функції, що має також цикл, тому загальна часова складність становить $O(n^2)$.

На останньому кроці визначається час виконання всіх робіт. У кожної машини підраховується тривалість роботи у циклі, після цього обирається час

виконання робіт, що є найбільшим. Складність цього кроку – $O(n^2)$, тому що у підрахунках наявний цикл із вкладеним циклом.

Асимптотична оцінка трудомісткості для алгоритму В за критерієм часу є $O(n^2)$ [3].

3.4 Приклад застосування розроблених алгоритмів

3.4.1 Постановка задачі для розв'язання

Підприємство «Балістика» отримало замовлення на виготовлення 16 виробів, час роботи над кожним з яких складає 1 або 2 години. Виготовлення кожного виробу можливе тільки на певній множині верстатів. Ці 16 робіт повинні бути розподілені між 4 верстатами.

Припускається, що дедлайни для виготовлення цих виробів не встановлені, а верстати не мають обмеження на кількість створених виробів.

Знайти розклад, за якого час завершення виготовлення останнього виробу (загальний час виготовлення всіх виробів) буде мінімальним.

Тривалості виготовлення та множини верстатів для виробів подані в таблиці 3.1.

Таблиця 3.1 – Тривалості виготовлення та множини верстатів виробів

Порядковий номер виробу	Тривалість виготовлення	Множина верстатів
1	1	1
2	1	4
3	1	2, 3
4	2	1, 2, 3
5	1	3
6	1	1, 2, 3, 4
7	2	2, 3
8	2	4
9	1	3
10	2	1, 2, 3
11	1	1
12	2	1
13	1	4
14	1	1, 2, 3, 4
15	2	1, 2, 3
16	2	4

3.4.2 Приклад розв’язання алгоритмом А

Для розв’язання задачі було отримано вхідні дані, а саме таблицю із тривалостями виготовлення виробів (таблиця 3.1).

Перш за все потрібно відсортувати роботи, за найменшою кількістю машин, на яких вона може виконуватись.

Відсортовані дані можна побачити в таблиці 3.2.

Таблиця 3.2 – Відсортовані вироби за множинами верстатів

Порядковий номер виробу	Тривалість виготовлення	Множина верстатів
1	1	1
2	1	4
5	1	3
8	2	4
9	1	3
11	1	1
12	2	1
13	1	4
16	2	4
3	1	2, 3
7	2	2, 3
4	2	1, 2, 3
10	2	1, 2, 3
15	2	1, 2, 3
6	1	1, 2, 3, 4
14	1	1, 2, 3, 4

Дані відсортовано, тому залишилося розставити роботи зі списку по машинах, враховуючи їх завантаженість.

На рисунках 3.1-3.3 показано перші три ітерації роботи алгоритму А, а на рисунку 3.4 отриманий фінальний розклад.

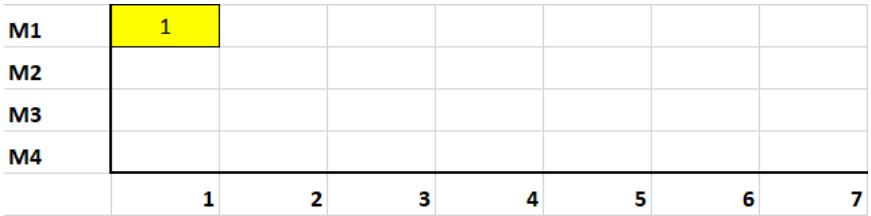

Рисунок 3.1 – Призначення першої роботи із списку на машину

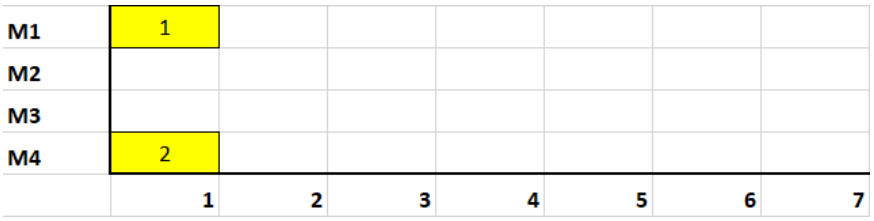

Рисунок 3.2 – Призначення другої роботи із списку на машину

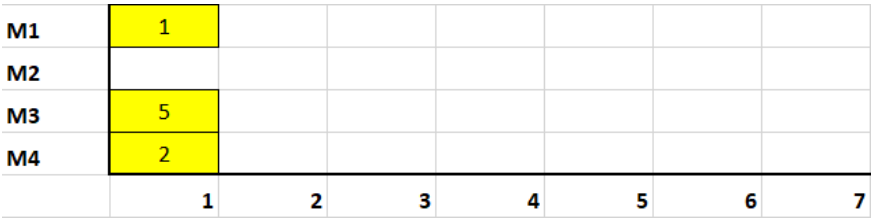

Рисунок 3.3 – Призначення третьої роботи із списку на машину

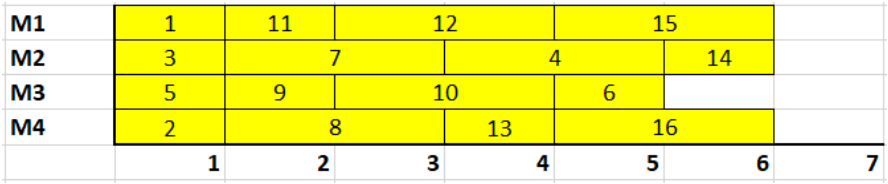

Рисунок 3.4 – Фінальний розклад

3.4.3 Приклад розв’язання алгоритмом B

Вхідні дані було отримано для подальшого розв’язання задачі, це таблиця, що містить тривалості виготовлення виробів (таблиця 3.1).

Треба розпочати із відсортування робіт спочатку за найменшою кількістю машин, де дана робота може виконуватись, після цього за найбільшою тривалістю.

Дані після відсортування знаходяться в таблиці 3.3.

Таблиця 3.3 – Відсортовані вироби за множинами верстатів та тривалостями виготовлення

Порядковий номер виробу	Тривалість виготовлення	Множина верстатів
8	2	4
12	2	1
16	2	4
1	1	1
2	1	4
5	1	3
9	1	3
11	1	1
13	1	4
7	2	2, 3
3	1	2, 3
4	2	1, 2, 3
10	2	1, 2, 3
15	2	1, 2, 3
6	1	1, 2, 3, 4
14	1	1, 2, 3, 4

Дані було відсортовано, залишилося розставити роботи, враховуючи їх завантаженість.

На рисунках 3.5-3.7 показано три перші ітерації роботи алгоритму B, а на рисунку 3.8 отриманий фінальний розклад.

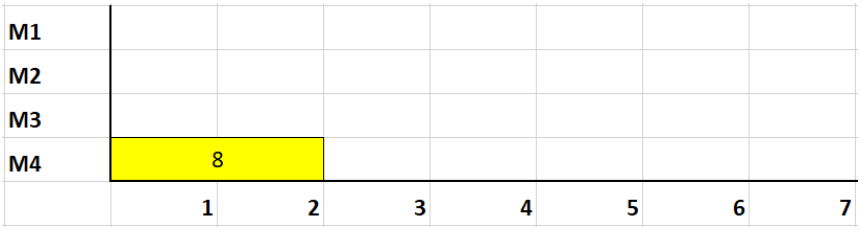

Рисунок 3.5 – Призначення першої роботи із списку на машину

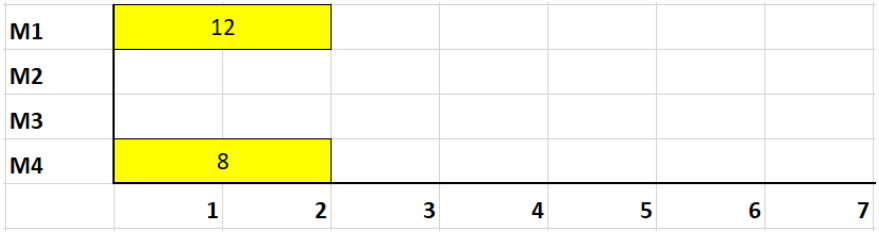

Рисунок 3.6 – Призначення другої роботи із списку на машину

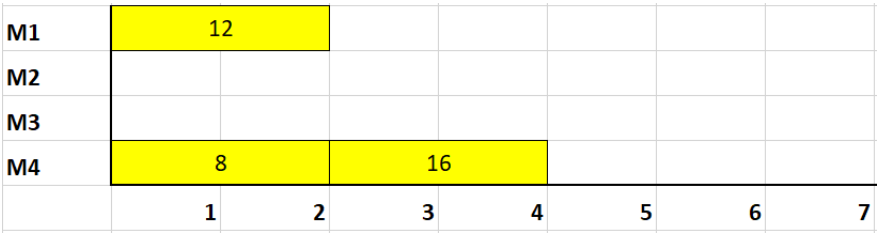

Рисунок 3.7 – Призначення третьої роботи із списку на машину

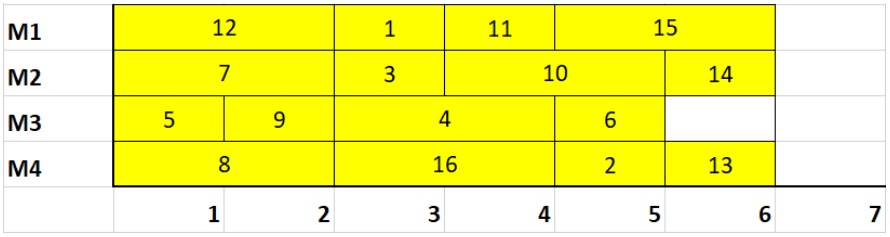

Рисунок 3.8 – Фінальний розклад

3.5 Планування експериментів

3.5.1 Класифікація індивідуальних задач

Робота алгоритмів надзвичайно залежать від особливостей вхідних даних задачі. Ефективність алгоритмів знаходження розкладу, за якого час завершення останньої роботи (загальний час виконання усіх робіт) буде мінімальним, залежить від впливу розмірності та абсолютних значень умовно-постійних параметрів, що подаються на вхід до індивідуальних задач (у нашому випадку це кількість машин, що працюють паралельно, кількість робіт та тривалість виконання робіт).

Отже, щоб дослідити алгоритми та порівняти їх ефективності, беручи різні початкові умови, необхідно виділити ключові ознаки, які допоможуть у класифікації індивідуальних задач (ІЗ).

Перелік загальних параметрів:

- m – кількість машин, що працюють паралельно;
- n – кількість робіт;
- p_i – тривалість виконання робіт, $p_i \in \{1,2\}$, де $i = \overline{1,n}$.

Кількість машин та кількість робіт ніяк не впливають на зміну ефективності роботи розроблених алгоритмів. Згідно з переліком залишених загальних параметрів алгоритму будемо класифікувати індивідуальні задачі за відносною кількістю робіт тривалістю 1 та 2.

3.5.2 План експериментів

Нижче наведено алгоритм дій для дослідження впливу тривалості виконання робіт на отримання оптимального значення цільової функції за допомогою алгоритму А та алгоритму В:

Початкова ініціалізація:

n:=100 //кількість робіт

m:=5 //кількість машин

// Створимо задачі 3 класів з різною тривалістю робіт;

// Перший клас матиме 75% робіт тривалістю 1, 25% - тривалістю 2

// Другий клас матиме 50% робіт тривалістю 1, 50% - тривалістю 2

// Третій клас матиме 25% робіт тривалістю 1, 75% - тривалістю 2

// Розв’яжемо задачі за допомогою алгоритму А та В та отримаєм результат

// Створимо 3 таблиці для задач трьох класів, які будуть містити результати алгоритмів

for i:=0 **to** 100 **do**: //згенерувати 100 задач і використати алгоритм А та В

GetProblem(n, m, 0.25) // згенерувати задачу з n робіт першого класу,

які виконуються на m машинах, що
працюють одночасно

Запустити алгоритм **A**

// *створити* рядок експерименту у таблиці для задач 1 класу та записати ЦФ

Запустити алгоритм **B**

// *додати* ЦФ у рядок експерименту

// *обрати* за ЦФ найкращий алгоритм для цього експерименту

GetProblem(n, m, 0.5) // згенерувати задачу з n робіт другого класу,
які виконуються на m машинах, що
працюють одночасно

Запустити алгоритм **A**

// *створити* рядок експерименту у таблиці для задач 2 класу та записати ЦФ

Запустити алгоритм **B**

// *додати* ЦФ у рядок експерименту

// *обрати* за ЦФ найкращий алгоритм для цього експерименту

GetProblem(n, m, 0.75) // згенерувати задачу з n робіт третього класу,
які виконуються на m машинах, що
працюють одночасно

Запустити алгоритм **A**

// *створити* рядок експерименту у таблиці для задач 3 класу та записати ЦФ

Запустити алгоритм **B**

// *додати* ЦФ у рядок експерименту

// *обрати* за ЦФ найкращий алгоритм для цього експерименту

end for

// Підрахувати в яких експериментах, який алгоритм дав кращий результат, та обчислити середнє відхилення ЦФ алгоритму від знайденого найкращого, вивести отримані результати

3.6 Порівняння алгоритмів за точністю

Оскільки жадібний алгоритм є наближеним, то він не гарантує отримання оптимального розв’язку. В якості оцінки множини допустимих розв’язків (нижньої границі значення цільової функції) було взято значення цільової функції за правилом LFJF.

Оскільки, алгоритм В є модифікацію алгоритму А і виконує сортування за тривалістю роботи, для порівняння їх розв’язків були створені три класи задач (описані у розділі 3.5.2).

Для експериментів були задані наступні параметри генерації задач: кількість робіт лежить в діапазоні від 100 до 1000 з кроком 10, кількість машин дорівнює 5. Коефіцієнт наближеності визначається як відношення оцінки множини допустимих розв’язків (значення цільової функції за правилом LFJF) до значення цільової функції фактично знайденого розв’язку; для оптимальних розв’язків такий коефіцієнт за визначенням становить 1.

Експерименти показали, що у всіх випадках алгоритм В знаходив оптимальний розв’язок. Алгоритм А у більшій частині випадків не знаходить оптимальний розв’язок. Це відбувається через те, що алгоритм не сортує роботи за тривалістю. Результати експериментів наведені на графіках на рисунках 3.9 - 3.11.

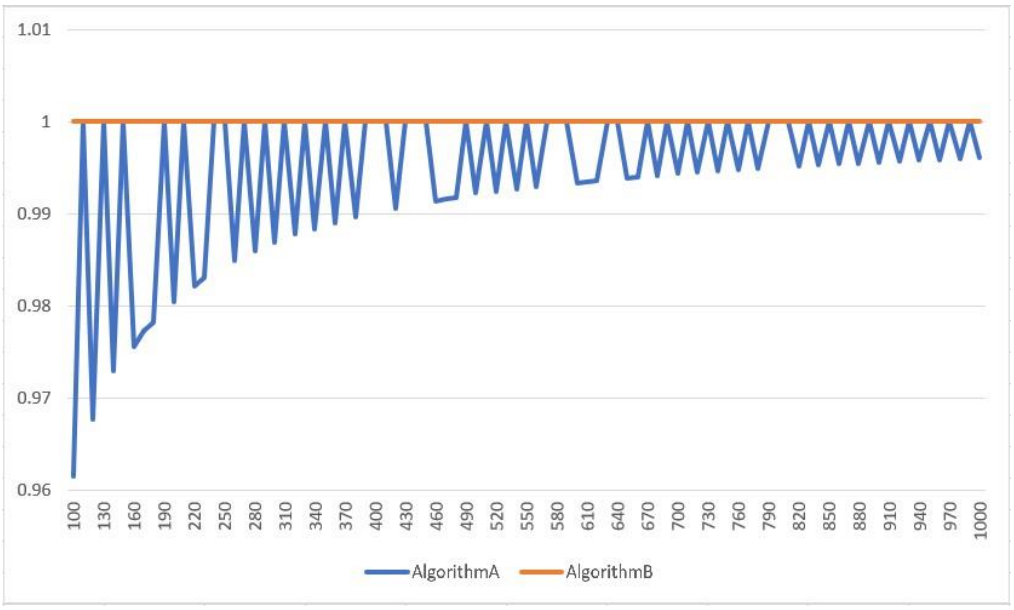

Рисунок 3.9 – Залежність коефіцієнта наближеності від кількості робіт (задача першого класу)

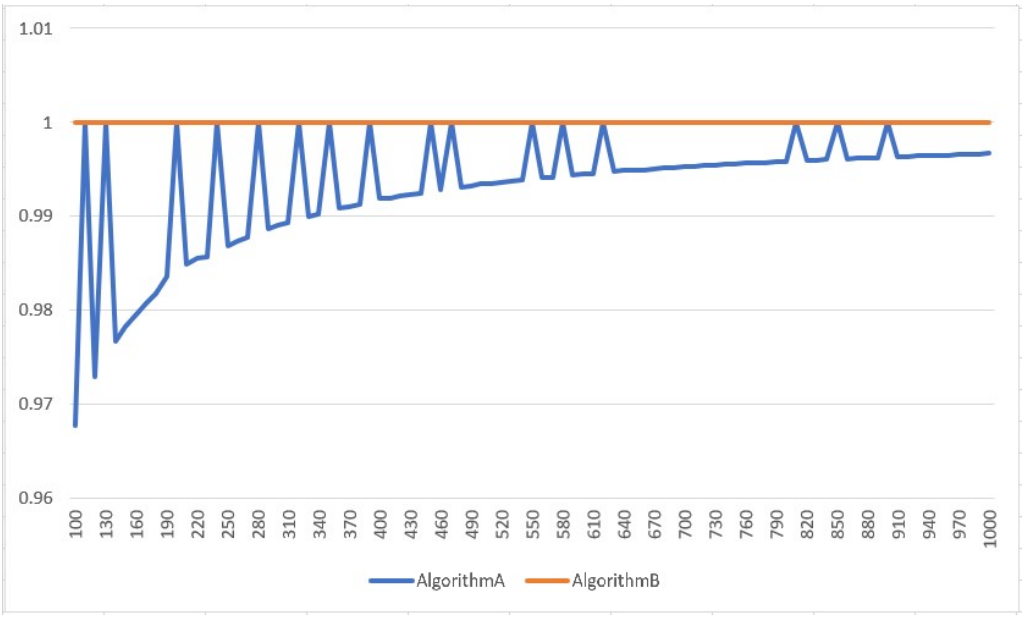

Рисунок 3.10 – Залежність коефіцієнта наближеності від кількості робіт (задача другого класу)

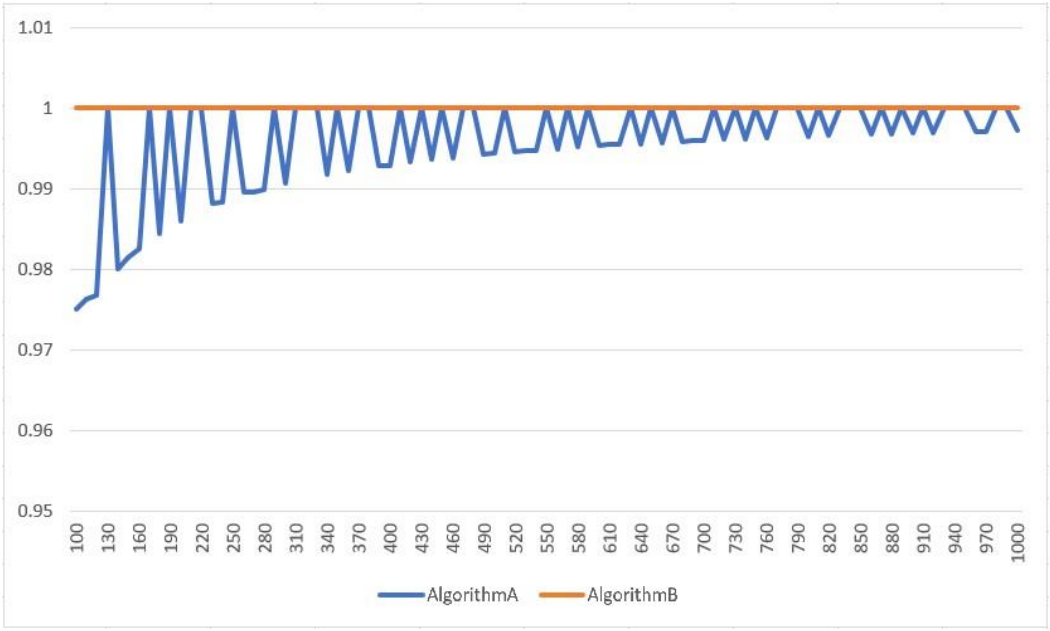

Рисунок 3.11 – Залежність коефіцієнта наближеності
від кількості робіт (задача третього класу)

Щоб більш наочно показати вплив важливості сортування за тривалістю був проведений додатковий експерименти з наступними параметрами: кількість робіт лежить в діапазоні від 100 до 1000 з кроком 10, кількість машин дорівнює 5, тривалість виконання робіт від 1 до 5. Результати експериментів наведені на графіку на рисунку 3.12.

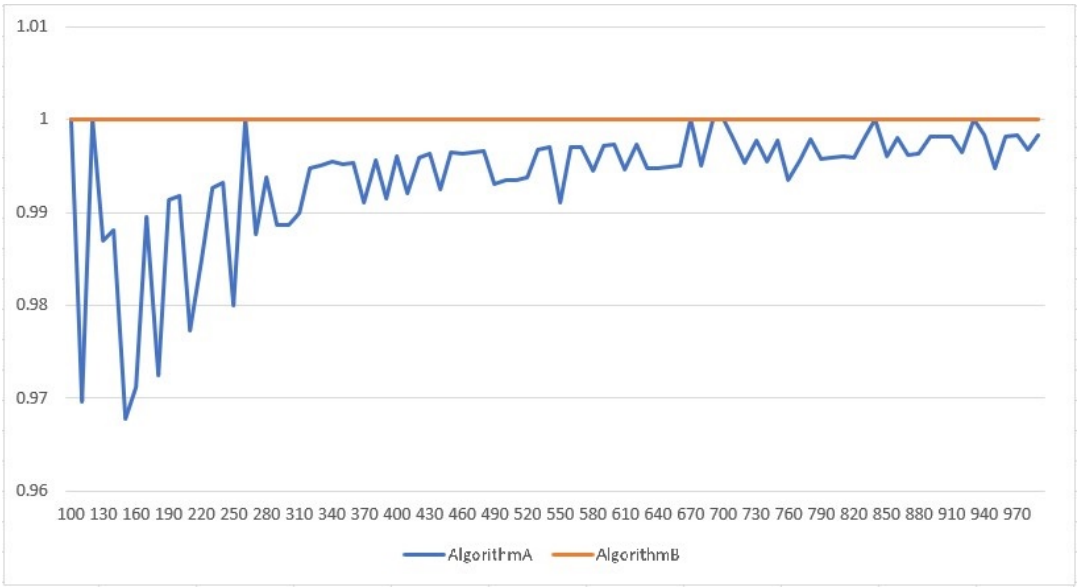

Рисунок 3.12 – Залежність коефіцієнта наближеності
від кількості робіт (тривалість робіт від 1 до 5)

Слід відзначити, що в даній серії експериментів значення коефіцієнта наближеності відмінне від 1 означає не знаходження розв’язку, далекого від оптимального, а знаходження розв’язку, значення цільової функції для якого перевищує нижню граничну оцінку. Ця різниця спричинена недостатньою кількістю машин або нерівномірним призначенням множин машин, на яких може виконуватись робота, при генерації задачі.

Відповідно до результатів експериментів, наведених на рисунках 3.9 - 3.11, алгоритм В у всіх випадках знаходить оптимальний, або не гірший за алгоритм А, розв’язок.

3.7 Порівняння алгоритмів за часовою складністю

Для практичної оцінки часової складності алгоритму за допомогою програмного продукту можуть бути проведені такі експерименти: для заданої кількості машин, в певному діапазоні кількості робіт із заданим кроком, генеруються по 20 задач заданої розмірності, а час їх розв’язання вимірюється у тактах процесора.

Для експериментів були задані наступні параметри генерації задач: кількість робіт лежить в діапазоні від 10 до 1000 з кроком 10, кількість машин дорівнює 5.

За результатами вимірювань був побудований графік, наведений на рисунку 3.13.

За графіком видно, що алгоритм LFJF [2], значно повільніший за його модифікації (алгоритм А та алгоритм В). Це пояснюється тим, що у правилі LFJF спочатку відбувається розбиття робіт з тривалістю більше 1 на роботи з тривалістю 1. Також, можна побачити, що алгоритм В працює повільніше за алгоритм А. Це відбувається через те, що в алгоритмі В виконується на одне сортування більше (сортування за тривалістю), ніж у алгоритмі А.

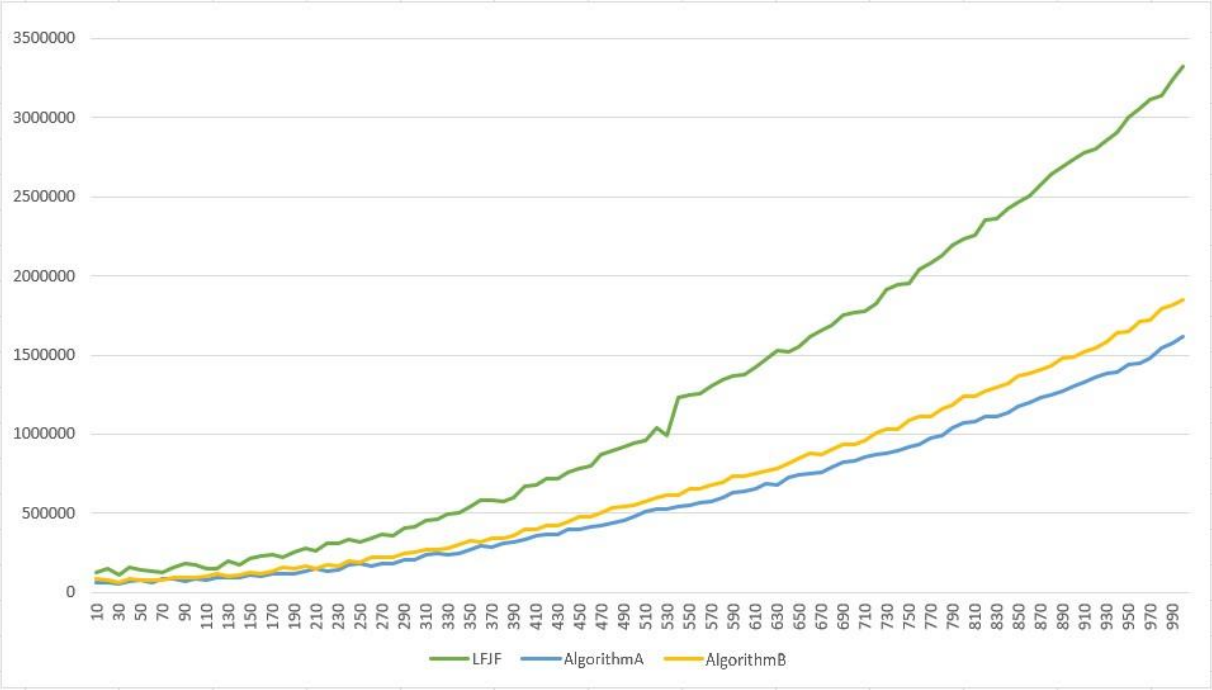

Рисунок 3.13 – Залежність середнього часу розв'язання задач варіаціями розробленого алгоритму від кількості робіт

На основі даних випробувань також була проведена оцінка часової складності алгоритму.

Був використаний такий підхід: перебір різноманітних можливих асимптотичних складностей та порівняння їх із залежністю часу виконання алгоритму від кількості робіт. Очевидно, що найкращою оцінкою складності буде така складність, за якої відношення залежності часу виконання алгоритму від кількості робіт до функції оцінки складності не буде суттєво змінюватися зі зміною розмірності вхідних даних (тобто матиме низьку дисперсію). У даному випадку такою функцією є $f(n)=n^2$, а відповідна часова складність – $O(n^2)$.

На рисунку 3.14 наведені порівняння залежностей часу розв'язання задач від їх розмірності та функцій $f_1(n)=k_1n^2$ та $f_2(n)=k_2n^2$, що пропорційні $f(n)=n^2$ з певними емпірично визначеними коефіцієнтами k_1 і k_2.

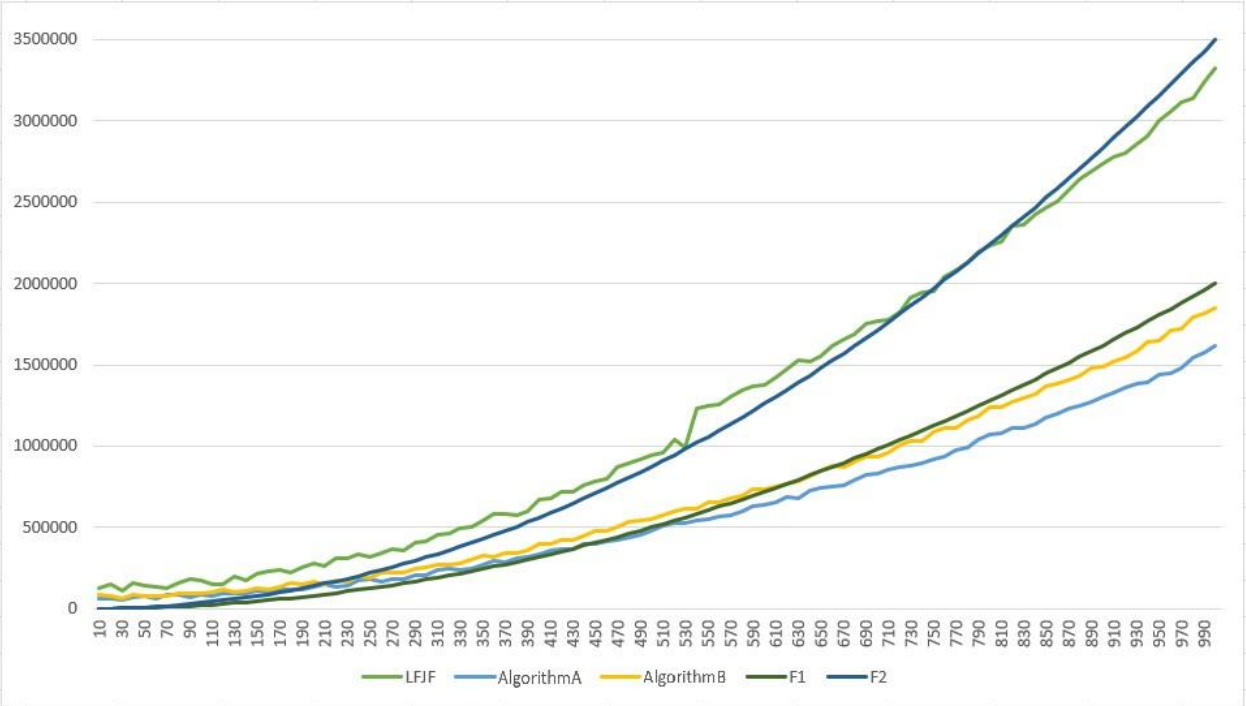

Рисунок 3.14 – Залежність середнього часу розв’язання задач варіаціями розробленого алгоритму від кількості робіт у порівнянні з функціями асимптотичної оцінки часової складності алгоритму

Висновок до розділу

У цьому розділі було сформовано математичну модель для даної індивідуальної задачі та розроблено два алгоритми для її розв’язання (подано як словесно так і у вигляді схем), було проведено серію експериментів над розробленими алгоритмами.

Оцінка часової складності розроблених алгоритмів – $O(n^2)$. Алгоритм А показав себе як найшвидший серед розроблених. Алгоритм В у всіх випадках знаходить оптимальний, або не гірший за алгоритм А, розв’язок.

4 ПРОГРАМНЕ ТА ТЕХНІЧНЕ ЗАБЕЗПЕЧЕННЯ

4.1 Опис засобів розробки

Для розробки програмного продукту було обрано мову програмування C#, а також фреймворк ASP.NET Core MVC [7], який використовується для створення веб-проектів.

Мова C# входить до платформи .NET Core, що є вільним та відкритим крос-платформним програмним забезпечення для побудови будь-яких програмних продуктів [6].

У проекті присутня реляційна база даних, було обрано систему управління реляційними базами даних під назвою MySQL, бо вона вже протягом багатьох років визнана однією з найкращих [10].

Для роботи із даними в базі даних використовується технологія програмування ORM (object-relational mapping). Найдосконалішою та найрозповсюдженішою ORM для мови програмування C#, є EF (Entity Framework) Core [8-9].

Кожне веб-застосування має на меті розміщення його у всесвітній глобальній мережі Інтернет. Воно може бути розміщено як on-cloud (на хмаровому веб-сервері) так і on-premise (на власному веб-сервері).

Для розміщення веб-застосування у всесвітній глобальній мережі Інтернет було обрано on-cloud підхід через високу доступність та гнучкість. Лідером в області хмарних обчислень є Amazon [11], тому його сервіси було обрано для цієї мети як оптимальні.

У проекті було застосовано наступні сервіси від Amazon: VPC (Virtual Private Cloud) [13], EC2 (Elastic Compute Cloud) [14], RDS (Relational Database Service) [15], Route 53 [12].

Для розробки використовувалось дві студії розробки – Visual Studio [16] та Visual Studio Code [17], а також система контролю версій Git [18].

Для повного розуміння призначення та цілей використання кожного із засобів розробки наводиться їх короткий опис:

– .NET Core – це нова версія .NET Framework, яка є безкоштовною універсальною платформою розробки з відкритим вихідним кодом, яку підтримує Microsoft. Це крос-платформний фреймворк, який працює на операційних системах Windows, macOS та Linux. C# входить до списку мов програмування, що підтримуються платформою .NET Core, та дозволяє розробникам створювати багато різноманітних безпечних і надійних застосувань, що використовуються у різних сферах IT;
– ASP.NET Core MVC – легка, з відкритим вихідним кодом, добре протестована платформа, оптимізована для використання з ASP.NET Core, що забезпечує спосіб створення динамічних веб-сайтів на основі шаблонів, що дозволяє чітко розділяти код продукту за його призначенням;
– MySQL – це система управління реляційними базами даних (RDBMS) з відкритим кодом Oracle на основі мови структурованих запитів (SQL). MySQL працює практично на всіх платформах, включаючи Linux, UNIX і Windows;
– Об’єктно-реляційне відображення (ORM) – це технологія програмування для перетворення даних між системами типів за допомогою об’єктно-орієнтованих мов програмування. Допомагає створити, по суті, «базу даних віртуальних об’єктів», яку можна використовувати мовами програмування;
– Entity Framework (EF) Core — це легка, розширювана, крос-платформна версія популярної технології доступу до даних Entity Framework. EF Core може служити об’єктно-реляційним інструментом, який дозволяє розробникам працювати з базою даних за допомогою об’єктів та усуває потребу в більшості коду доступу до

даних, який зазвичай потрібно написати. EF Core підтримує багато механізмів баз даних.

4.2 Вимоги до технічного забезпечення

Технічні вимоги для локального запуску програмного продукту, які необхідні для його очікувано коректної роботи:

- оперативна пам’ять – 8 ГБ або більше;
- жорсткий диск – 50ГБ або більше;
- встановлена платформа .NET Core 5.0;
- встановлена СУБД MySQL.

Єдиною технічною вимогою для віддаленого підключення до програмного продукту у мережі інтернет є стабільне підключення до інтернету.

4.3 Архітектура програмного забезпечення

4.3.1 Схема архітектури програмного забезпечення

Для розробки програмного продукту було застосовано патерн проектування MVC та багатошарову архітектуру проєктування, через це веб-застосування та логіка роботи алгоритмів розбита на два проєкти.

Візуальна складова програмного продукту зберігається у проекті WebApplication, а саме у папках:

- Models (Моделі) – проміжні сутності для передачі у представлення;
- Views (Представлення) – сторінки, які відображають передані із контролерів дані;
- Controllers (Контролери) – класи, що містять методи із логікою для відображення представлень та передачею даних у них.

Основні файли логіки містяться у проекті Scheduling у наступних папках:

– Data (Дані) – індивідуальна задача із курсової;
– Entities (Сутності) – основні сутності для вирішення задач;
– Helpers (Помічники) – класи для спрощення роботи з даними;
– Solvers (Розв’язувачі) – алгоритми, які використовуються при розв’язанні.

Схема структурна патерну MVC із структурою файлів розробленого програмного продукту представлена на рисунку 4.1.

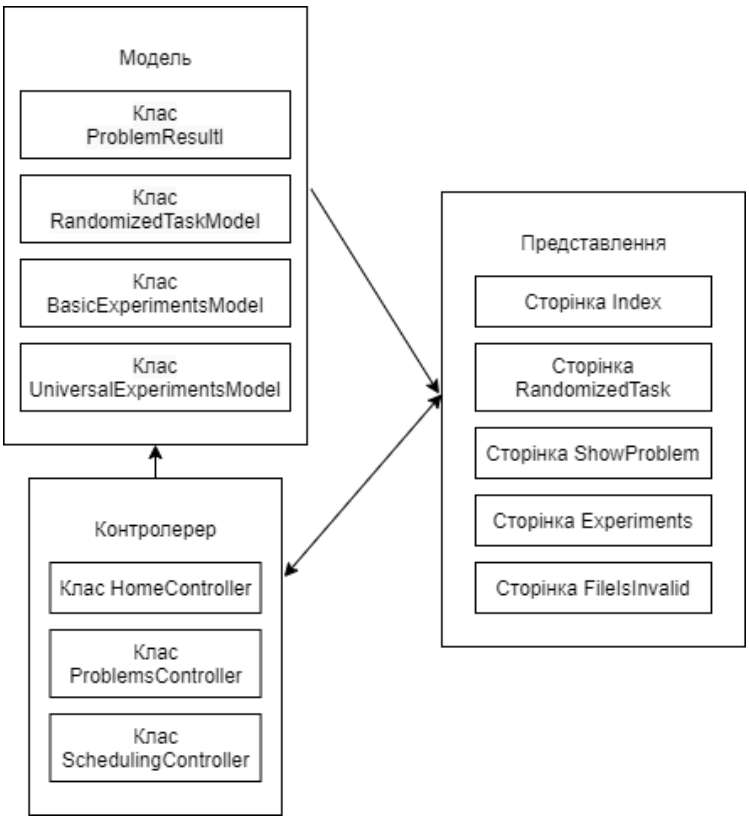

Рисунок 4.1 – Схема структурна патерну MVC із структурою файлів програмного продукту

Було обрано on-cloud підхід для розгортання веб-застосування у мережі Інтернет. AWS (Amazon Web Services) було використано для досягнення цієї мети.

Веб-проєкт та база даних знаходяться у одній зоні доступу, попри те базу даних було винесено у приватну мережу, щоб обмежити неправомірний

доступ до неї. Веб-проєкт розміщено у публічній мережі, він має доступ до приватної мережі, щоб читати та записувати дані у базу даних.

За допомогою сервісів Internet Gateway та Route 53 доступ до застосування можливий за обраним доменним ім’ям, що полегшує його пошук у мережі Інтернет.

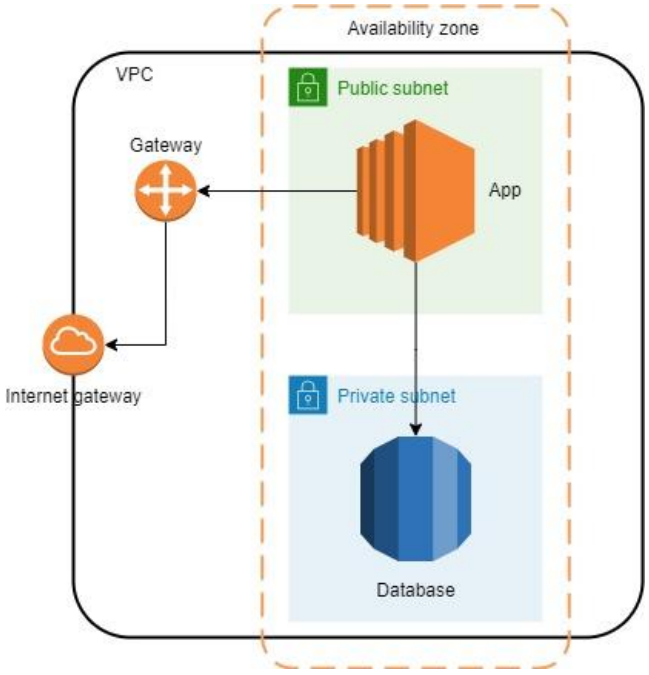

Рисунок 4.2 – Схема структурна розгортання програмного продукту у мережі Інтернет за допомогою сервісів AWS

4.3.2 Діаграма класів

На рисунку 4.3 наведено діаграму класів сутностей із бібліотеки класів Scheduling. Ці класи використовуються алгоритмами для вирішення задач та повернення результатів у контролер для подальшого їх відображення на сторінці користувача.

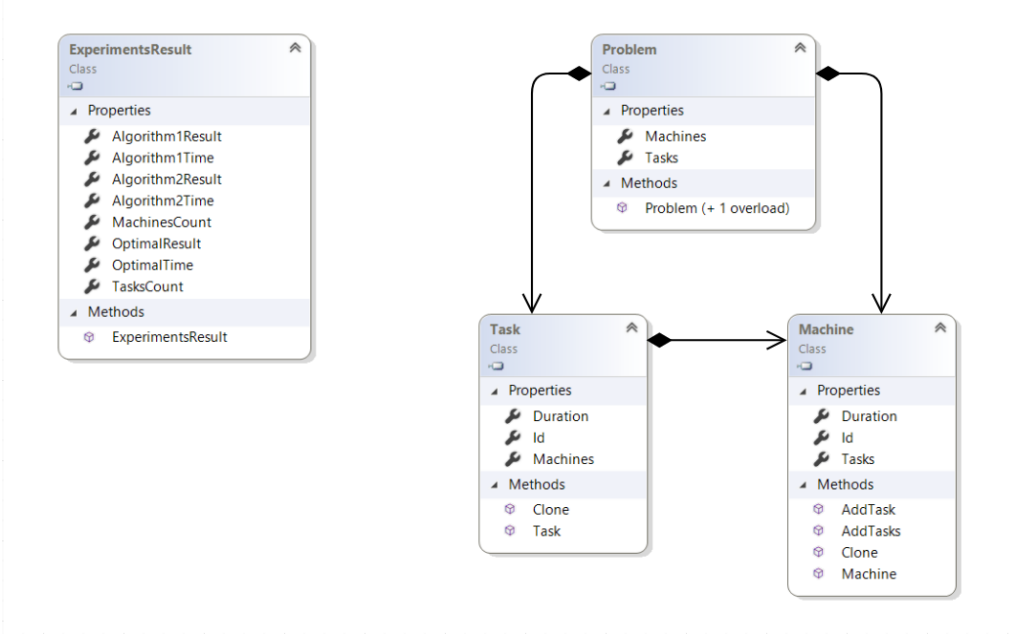

Рисунок 4.3 – Схема структурна класів проєкту Scheduling

На рисунку 4.4 наведено діаграму класів сутностей із бібліотеки класів WebApplication. Ці класи використовуються контролерами для отриманнями даних від користувача для генерації задач, їх подальшого розв'язання та відображення на сторінці користувача.

Рисунок 4.4 – Схема структурна класів проєкту WebApplication

4.3.3 Діаграма діяльності

Графічні матеріали містять наступні діаграми діяльності варіантів використання:

- «Розв’язання задачі за даними на вхід, що згенеровані випадково»;
- «Розв’язання задачі за даними на вхід, що зберігаються у файлі»;
- «Вимірювання значення цільових функцій у задачах, що мають різну розмірність та тривалість виконання робіт 1 та 2»;
- «Вимірювання значення цільових функцій у задачах, що мають різну розмірність та будь-яку тривалість виконання робіт».

4.3.4 Специфікація функцій

За логіку відображення даних відповідають контролери, у веб-застосунку WebApplication присутні такі контролери: HomeController, SchedulingController, ProblemsController.

У класі HomeController містяться наступні публічні методи:

- Index() – відображає представлення із головним меню застосування, на якому можна обрати бажаний сценарій роботи;
- Error() – викликається системно при критичній помилці у роботі програми.

У класі SchedulingController містяться наступні публічні методи:

- FileUpload() – приймає на вхід файл із завданням, перевіряє його, складає розклад, і модифікує відповідний екземпляр класу Problem таким чином, щоб він зберігав інформацію про розклад, повертає сторінку із розкладами розробленими усіма алгоритмами;
- RandomizedTask() – приймає на вхід параметри для генерації індивідуальної задачі, складає розклад, і модифікує відповідний екземпляр класу Problem таким чином, щоб він зберігав інформацію

про розклад, повертає сторінку із розкладами розробленими обраними алгоритмами;

- BasicExperiments() – приймає на вхід параметри для генерації індивідуальних задач із довжиною виконання робіт 1 та 2, складає розклади, зберігає значення ЦФ, повертає сторінку із значеннями ЦФ отриманими обраними алгоритмами;
- UniversalExperiments() – приймає на вхід параметри для генерації індивідуальних задач із будь-якою довжиною виконання робіт, складає розклади, зберігає значення ЦФ, повертає сторінку із значеннями ЦФ отриманими обраними алгоритмами.

У класі ProblemsController містяться наступні публічні методи:

- ShowProblem() – відображає представлення із останньою згенерованою індивідуальної задачею, а саме всі роботи, які входять до неї, надає можливість зберегти їх у файл;
- SaveToFile() – завантажує останню згенеровану індивідуальну задачу на комп’ютер користувача у якості текстового фалу із розширенням txt та назвою problem.

Висновок до розділу

У цьому розділі описані технології, що були обрані для подальшої програмної реалізації індивідуального дослідницького проєкту. Наведено технічні вимоги для реалізації у програмному продукті. Діаграму класів було наведено так само, як специфікації основних функцій коду у програмному продукті.

5 ТЕХНОЛОГІЧНИЙ РОЗДІЛ

5.1 Керівництво користувача

Користувач потрапляючи на початкову сторінку сайту призначеного для складання розкладів (рисунок 5.1) отримає можливість самостійно вирішити, який варіант введення чи генерації вхідних даних для подальшого розв’язання задачі йому підходить найбільше. При виборі проведення експериментів користувачу так само надається широкий вибір опцій для задання параметрів індивідуальної задачі.

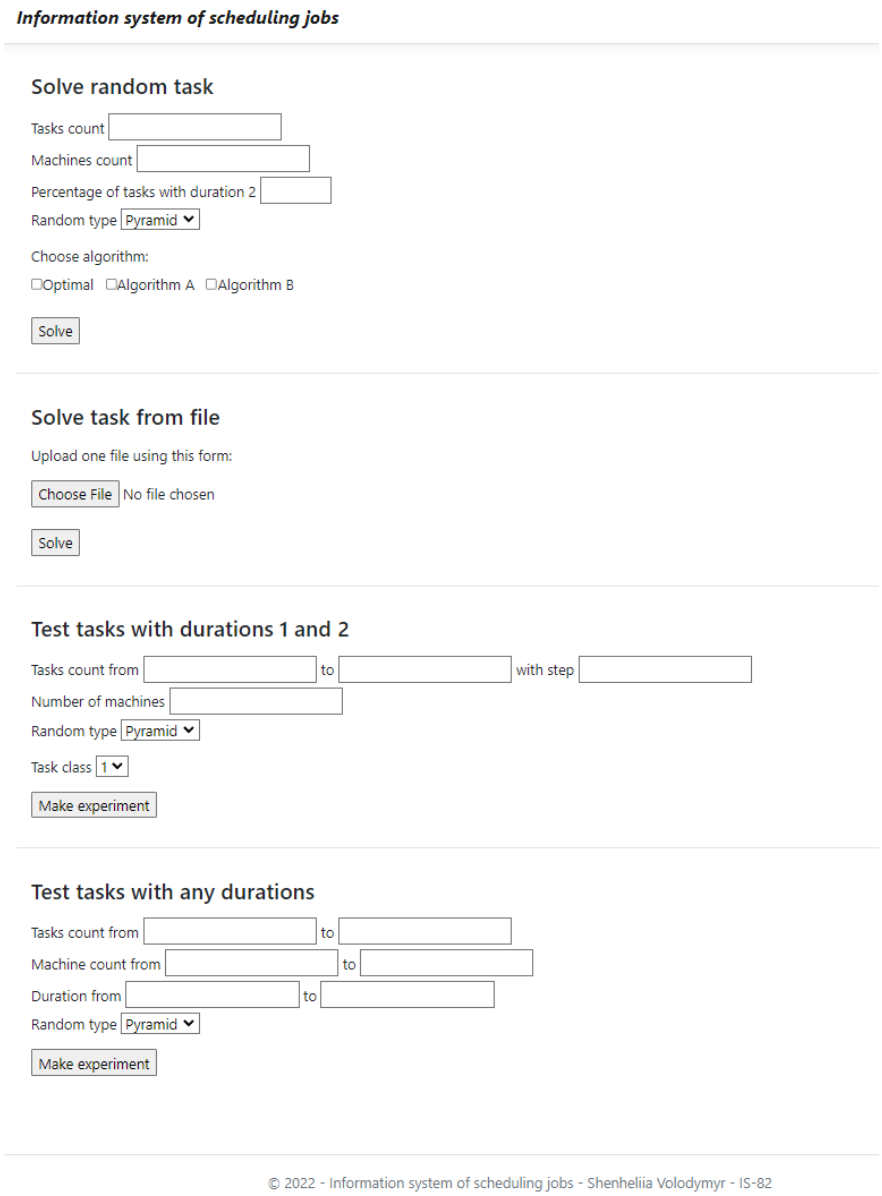

Рисунок 5.1 – Початкова сторінка програмного продукту

Якщо користувач зробить вибір варіанту використання «Розв’язання задачі за даними на вхід, що згенеровані випадково», то перед ним постане необхідність у введені розмірності задачі, а саме кількості робіт і кількості машин, виборі відсотку робіт із довжиною 2 та типу генерації вкладених множин робіт, виборі бажаних варіацій алгоритмів, після чого потрібно натиснути Solve.

В результаті цих дій користувача генеруються дані, що подаються на вхід. Розклад буде складено на основі попередньо згенерованих даних, після чого він буде виведений на екран у вигляді діаграми Ґанта (рисунок 5.3).

На згенерованій сторінці із розв’язком індивідуальної задачі можна натиснути кнопку Show problem, після чого буде показано згенеровану індивідуальну задачу (рисунок 5.4), де на сторінці ще буде кнопка Save to file, що надає можливість зберегти умову задачі у файл (рисунок 5.5).

Приклад введення параметрів у форму (кількість робіт – 20; кількість машин – 6; відсоток робіт із довжиною 2 – 0.6; обраний тип генерації вкладених множин робіт – Pyramid; алгоритм не обрано, тому задача буде розв’язана за допомогою усіх трьох розроблених алгоритмів) наведено на рисунку 5.2.

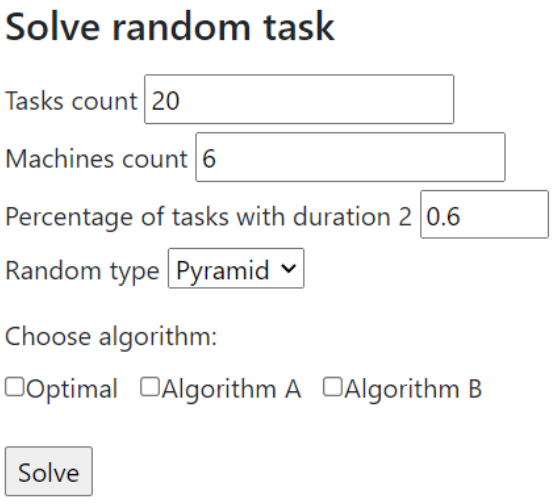

Рисунок 5.2 – Приклад форми із введеними параметрами

Результат саме генерації розкладу за даними на вхід, які згенеровані випадково, подано на рисунку 5.3.

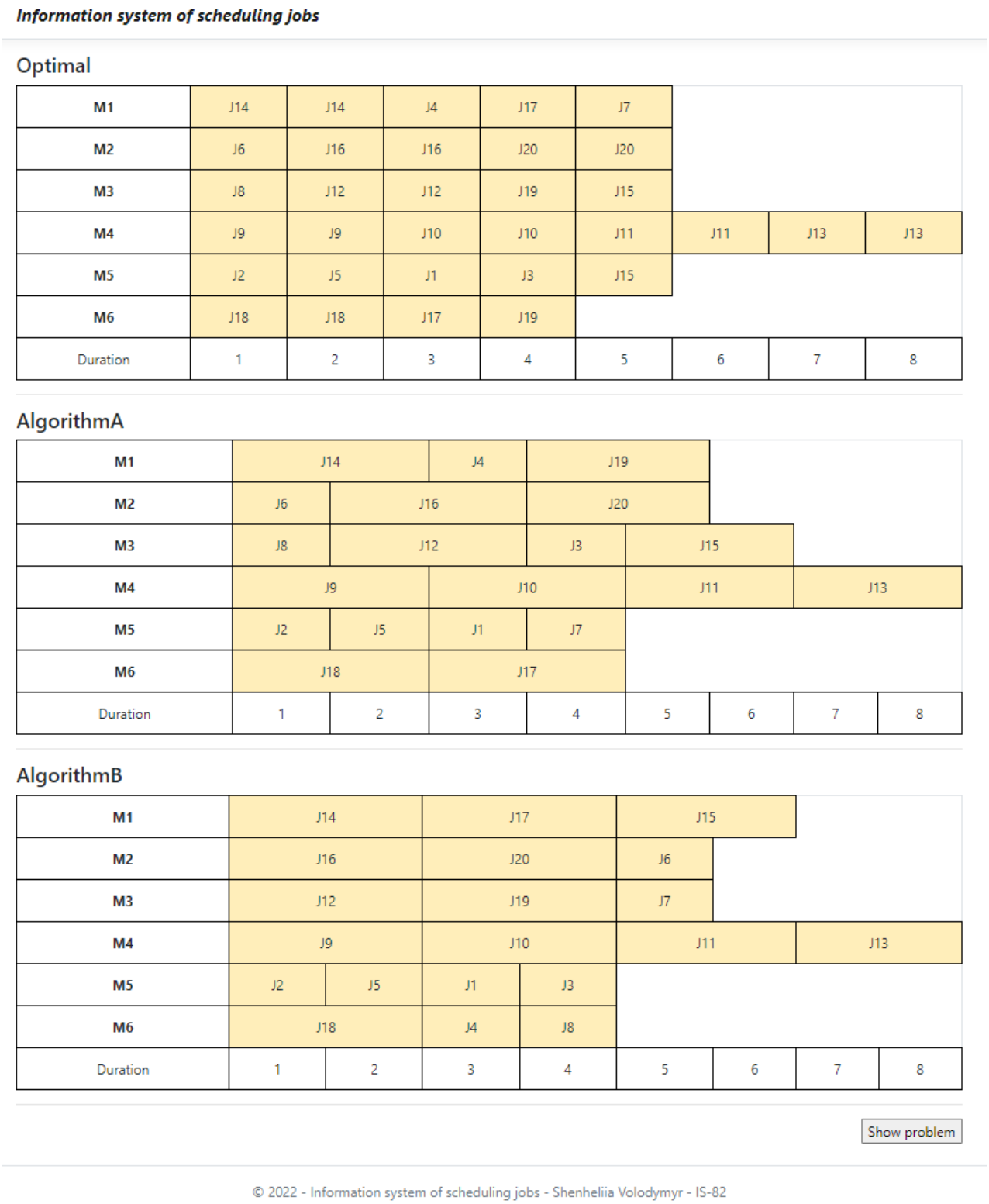

Рисунок 5.3 – Створений розклад за даними згенерованими випадково

На рисунку 5.4 зображено сторінку із згенерованою задачею, а рисунок 5.5 відображає частину збереженої у файл згенерованої задачі.

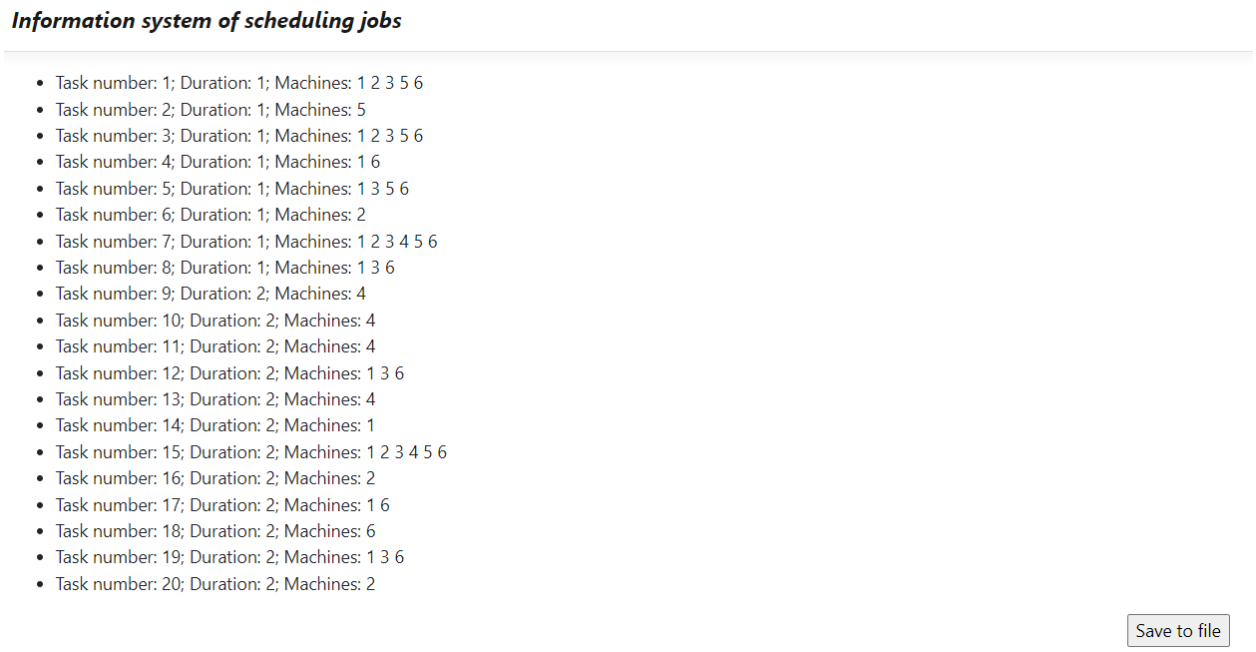

Рисунок 5.4 – Сторінка із згенерованою задачею

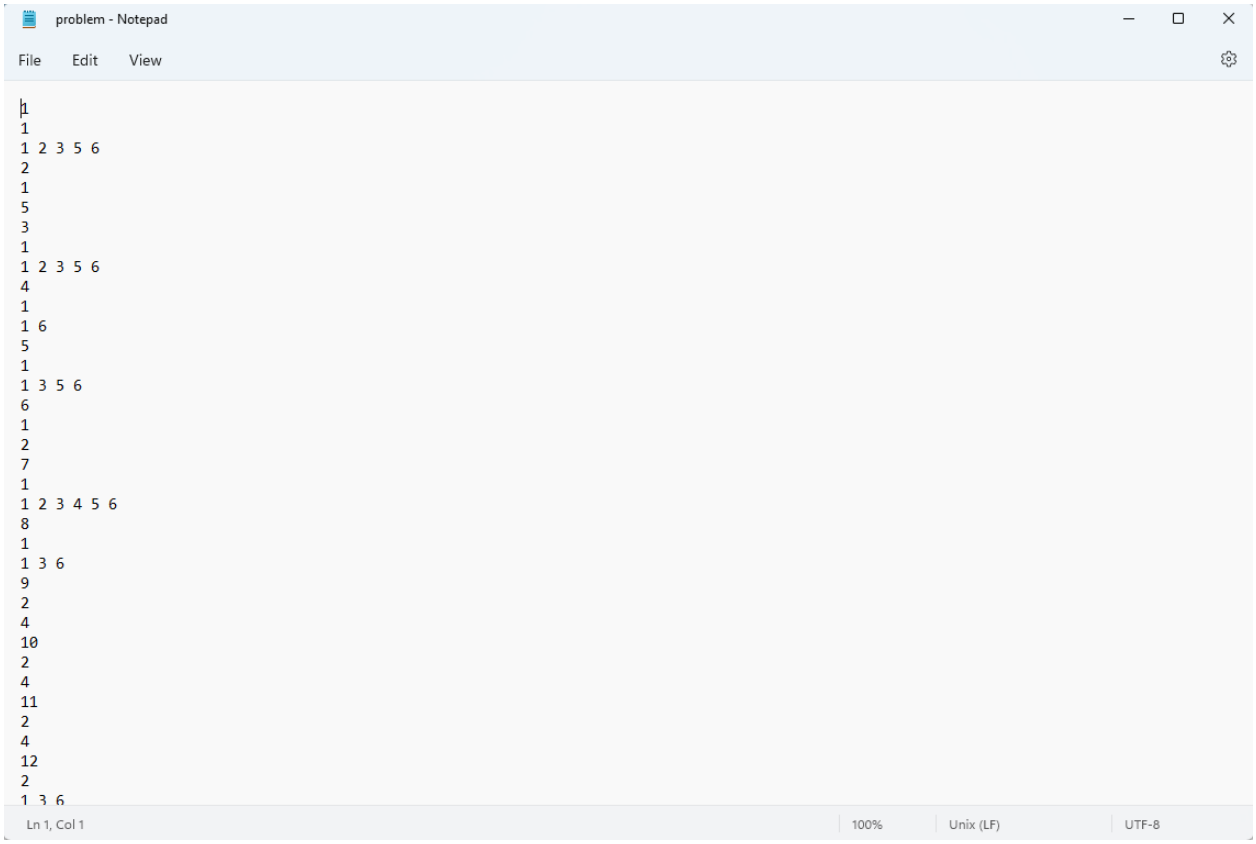

Рисунок 5.5 – Частина збереженої у файл згенерованої задачі

Якщо користувач зробить вибір варіанту використання «Розв’язання задачі за даними на вхід, що зберігаються у файлі», то тоді перед ним постане необхідність у створенні файлу за наступним правилом: кожні три рядки файлу визначають роботу, де перший рядок – номер роботи, другий – тривалість її виконання, третій – машини, на яких вона може виконуватись (більш детально описано у розділі 2.1).

Завантажити створений файл потрібно через форму, яка знаходиться на початковій сторінці сайту, натиснувши кнопку «Обрати файл», після цього натиснути Solve, розклад буде згенеровано на сторінці (рисунок 5.7) чи з’явиться повідомлення, що виведе інформацію про помилку (рисунок 5.8), якщо структура вхідних даних у файлі була порушена.

Приклад форми із передачею вхідних даних у вигляді файлу (назва файлу – problem.txt) наведено на рисунку 5.6.

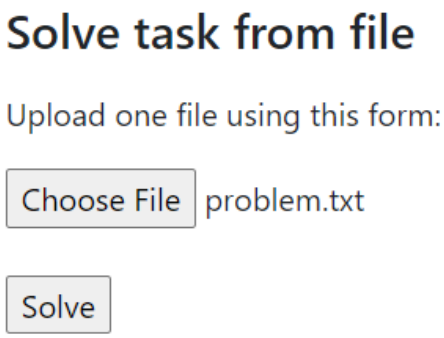

Рисунок 5.6 – Приклад форми із введеними параметрами

Результат саме генерації розкладу за даними на вхід, які передано у вигляді файлу, подано на рисунку 5.7.

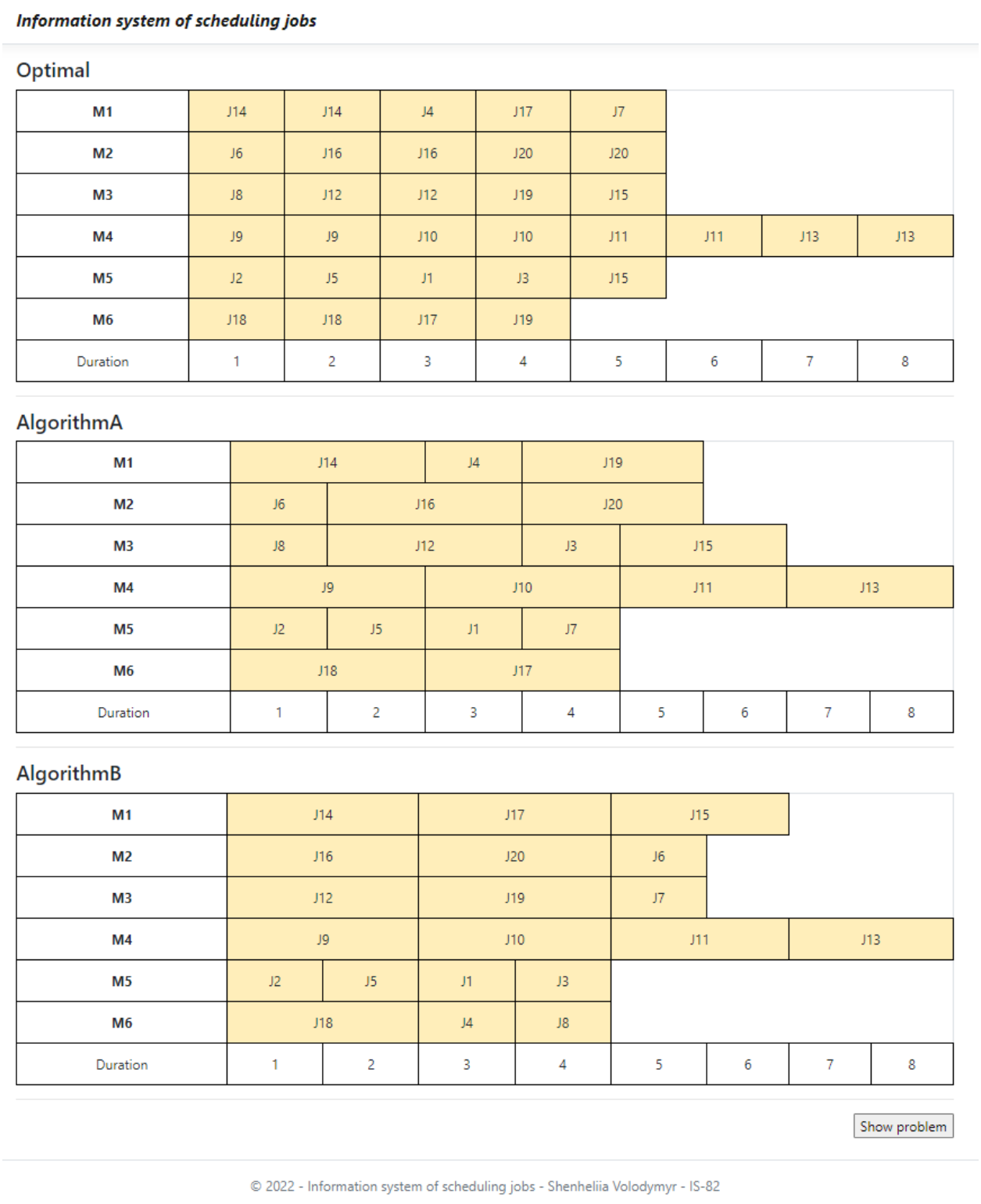

Рисунок 5.7 – Створений розклад за даними, які передано у вигляді файлу

На рисунку 5.8 показано сторінку, яка відображається, якщо структура робіт у файлі була порушена.

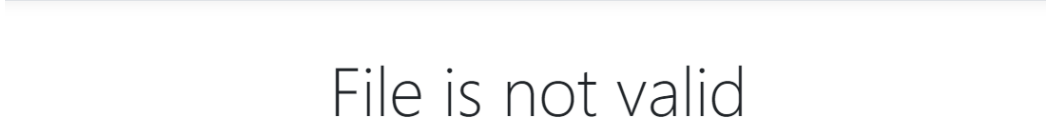

Рисунок 5.8 – Приклад повідомлення про генерацію розкладу на основі пошкоджених даних із файлу

Можна побачити, що розклад на рисунках 5.3 та 5.7 співпадає, це означає, що для певної індивідуальної задачі завжди будується однаковий розклад.

Якщо користувач зробить вибір варіанту використання «Вимірювання значення цільових функцій у задачах, що мають різну розмірність та тривалість виконання робіт 1 та 2», то тоді перед ним постане необхідність у введені сталої кількості машин та діапазону кількості робіт, а також кроку між кількістю робіт, за яким буде обиратися розмірність, обрати тип генерації вкладених множин робіт та відсоток робіт із довжиною 2, після чого натиснути Make experiment.

У результаті цих дій вхідні дані будуть генеруватися автоматично. На основі згенерованих даних складається розклад і наприкінці виводиться на сторінку.

На рисунку 5.9 можна побачити введені дані для експерименту, за якими програма випадковим чином згенерує 50 задач за заданими розмірностями.

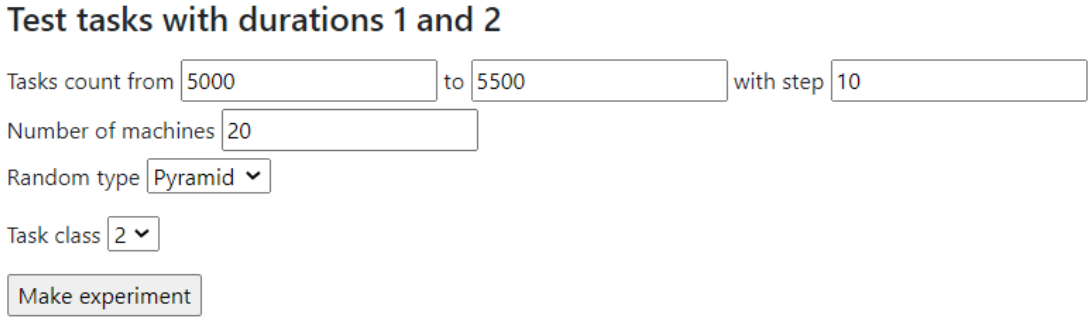

Рисунок 5.9 – Приклад форми із введеними параметрами

Згенеровані задачі розв’язуються за допомогою трьох алгоритмів, цільова функція кожної індивідуальної задачі записуються у таблицю, частину якої можна побачити на рисунку 5.10.

Tasks	Machines	Optimal result	Algorithm A result	Algorithm B result
5000	20	378	379	378
5010	20	378	379	378
5020	20	377	378	377
5030	20	379	380	379
5040	20	378	379	378
5050	20	379	380	379
5060	20	383	383	383
5070	20	382	382	382
5080	20	384	385	384
5090	20	382	383	382
5100	20	384	385	384
5110	20	384	384	384
5120	20	385	385	385
5130	20	388	388	388
5140	20	386	386	386
5150	20	387	387	387
5160	20	387	388	387
5170	20	391	391	391
5180	20	392	392	392
5190	20	392	392	392
5200	20	393	393	393
5210	20	392	392	392
5220	20	392	392	392

Рисунок 5.10 – Таблиця зі значеннями цільової функції для кожної індивідуальної задачі з тривалістю робіт 1 чи 2

Час розв’язання кожної задачі для усіх алгоритмів вимірюється у мілісекундах. Середній час розв’язання задач, що мають однакову розмірність, також знаходиться на цій сторінці окремо за кожним алгоритмом.

Також на сторінці виводиться відхилення від «оптимального» значення, кількість разів, коли алгоритм знайшов оптимальне значення. Ці результати можна побачити на рисунку 5.11.

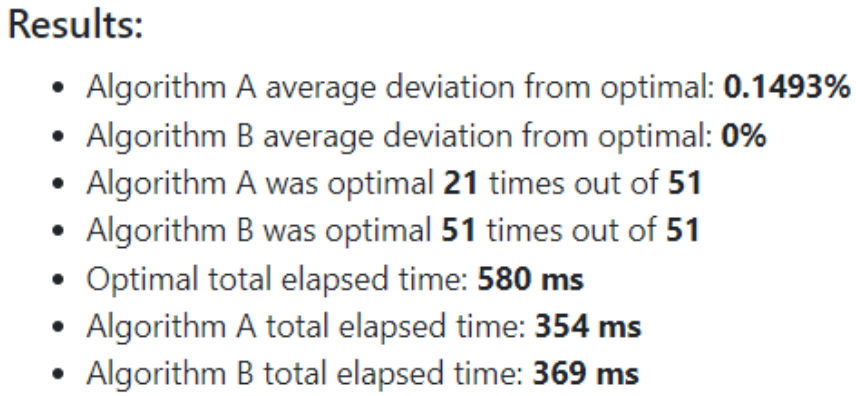

Рисунок 5.11 – Результати експерименту

Якщо користувач зробить вибір варіанту використання «Вимірювання значення цільових функцій у задачах, що мають різну розмірність та будь-яку тривалість виконання робіт», то тоді перед ним постане необхідність у введені діапазонів кількості машин та кількості робіт, а також діапазону тривалості робіт, обрати тип генерації вкладених множин робіт, після чого натиснути Make experiment.

На рисунку 5.12 можна побачити введені дані для експерименту, за якими програма випадковим чином створить 14 задач за заданими розмірностями.

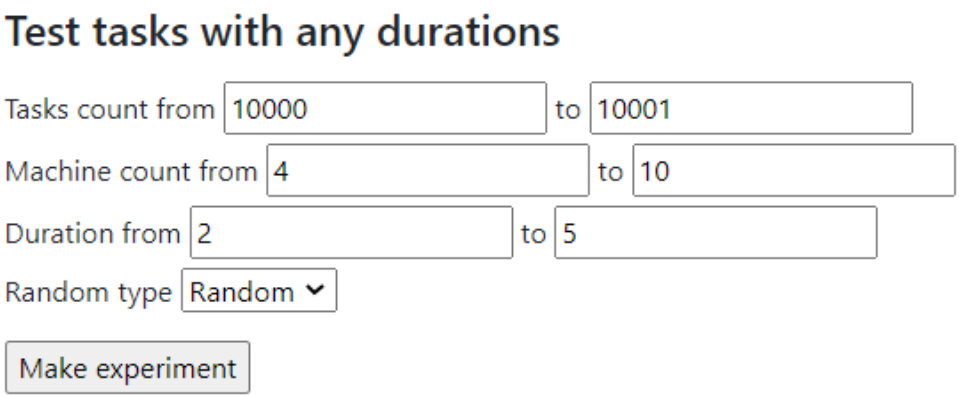

Рисунок 5.12 – Приклад форми із введеними параметрами

Як і в попередньому випадку із проведенням експериментів згенеровані задачі будуть розв'язуються трьома алгоритмами, а цільова функція для кожної індивідуальної задачі також буде записана в таблицю, яка знаходиться на рисунку 5.13.

Tasks	Machines	Optimal result	Algorithm A result	Algorithm B result
10000	4	8701	8702	8702
10000	5	7002	7003	7002
10000	6	5825	5826	5825
10000	7	5058	5060	5059
10000	8	4402	4405	4402
10000	9	3888	3890	3888
10000	10	3479	3481	3480
10001	4	8720	8721	8720
10001	5	7021	7022	7021
10001	6	5811	5812	5811
10001	7	5015	5016	5015
10001	8	4395	4397	4395
10001	9	3879	3880	3879
10001	10	3482	3484	3483

Рисунок 5.13 – Таблиця зі значеннями цільової функції
для кожної індивідуальної задачі з довільною тривалістю робіт

Так само як і в попередньому експерименті на сторінці знаходиться інформація про відхилення від «оптимального» значення та інші аналогічні показники, що можна побачити на рисунку 5.14.

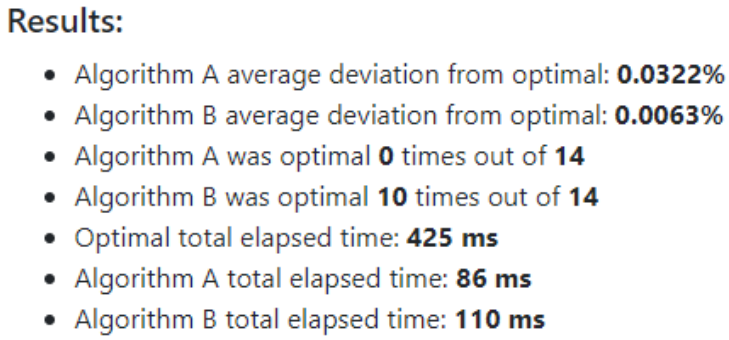

Рисунок 5.14 – Результати експерименту

5.2 Випробування програмного продукту

5.2.1 Мета випробувань

За мету випробувань поставлено перевірку відповідності функціоналу інформаційної системи з підтримка дослідження оптимізаційної задачі із теорії розкладів, а саме задачі складання розкладів, коли роботи виконуються паралельними машинами, для яких наявна умова вкладеності, до вимог технічного завдання.

5.2.2 Загальні положення

Наступні документи було взято за основу для випробувань:

– ГОСТ 34.603–92. Інформаційна технологія. Види випробувань автоматизованих систем;

– ГОСТ РД 50-34.698-90. Автоматизовані системи вимог до змісту документів.

5.2.3 Результати випробувань

Для кожного з варіантів використання, а саме «Розв’язання задачі за даними на вхід, що згенеровані випадково», «Вимірювання значення цільових функцій у задачах, що мають різну розмірність та тривалість виконання робіт 1 та 2», «Вимірювання значення цільових функцій у задачах, що мають різну розмірність та будь-яку тривалість виконання робіт», тести не були застосовані, оскільки вхідна задача генерується автоматично відповідно до правил, що виключають можливість виникнення помилки.

При введенні розмірності помилка також вважається виключеною, оскільки форма налаштована таким чином, що до відповідних полів можна вводити лише цілі числа, що більше або дорівнюють 2.

У той час для варіанта використання «Розв’язання задачі за даними на вхід, що зберігаються у файлі» були застосовані як позитивні так і негативні тести.

Позитивні тести представляють собою випадок, у якому на вхід буде передано текстовий файл, що містить інформацію у коректній формі, у такому випадку програма мусить працювати відповідно до очікуваного, а саме скласти розклад робіт для індивідуальної задачі за вхідними даними.

Негативні тести представляють собою випадок, у якому на вхід буде передано текстовий файл, що містить інформацію у некоректній формі, у такому випадку програма мусить вивести відповідне повідомлення про помилку.

Під час проведення випробувань програмного продукту були успішно пройдені усі позитивні та негативні тести.

Висновок до розділу

У даному розділі було детальне описано керівництво користувача для користування веб-застосуванням призначеним для складання розкладів за вхідними даними індивідуальних задач досліджуваної задачі.

Приклади екранних форм було представлено для перегляду, а також було наведено інформацію щодо варіантів використання за кожною із них окремо.

Наведено опис випробувань, що були проведені над програмним продуктом, які націлені на пересвідчення у коректності перевірки вхідних даних як для випадково згенерованих задач, так і для задач із вхідними даними, що були подані у вигляді файлу.

ЗАГАЛЬНІ ВИСНОВКИ

Під час роботи над індивідуальним дослідницьким проєктом було розроблено інформаційну систему з підтримки дослідження задачі складання розкладу виконання робіт паралельними машинами, для яких виконується умова вкладеності.

Розроблений програмний продукт має функціонал, що забезпечує передачу вхідних даних декількома варіантами для подальшого дослідження індивідуальної задачі (генерація випадкових даних за поставленою розмірністю задачі та завантаження умови задачі за допомогою файлу, що відповідає поставленій структурі).

Розділ загальних положень описує сформульовану змістовну постановку задачі, дослідження розв'язання якої реалізовано у вигляді програмного продукту. Також наведено приклад практичної індивідуальної задачі та певних допустимих її розв'язків. Задача є новою, тому аналогів програмних продуктів до створеного не існує.

У розділі інформаційного забезпечення описано вхідні та вихідні данні, структура бази даних та їх використання у процесі створення програмного продукту.

У розділі математичного забезпечення була сформована математична модель для поставленої задачі, два алгоритми було розроблено для її розв'язання (подані словесно та у вигляді схем). Серія експериментів для перевірки розроблених алгоритмів була проведена, у результаті чого було виявлено, що алгоритм А можна вважати за найшвидший серед розроблених. У той час як за допомогою алгоритму В у всіх випадках було отримано оптимальний, або не гірший за алгоритм А, розв'язок. Часова оцінка складності розроблених алгоритмів становить $O(n^2)$.

Розділ програмного та технічного забезпечення описує технології, що були обрані для подальшої програмної реалізації індивідуального

дослідницького проєкту. Наведено технічні вимоги для реалізації програмного продукту. Діаграма класів була наведена так само, як і таблиця специфікації основних функцій коду у програмному продукті.

Технологічний розділ спрямовано на детальний опис керівництва користувача для веб-застосування з прикладами екранних форм. Також було наведено перелік випробувань, усі випробування були проведені, програмний продукт працює відповідно до поставлених вимог.

Усі вимоги до розробки програмного продукту із технічного завдання було виконано. Модуль, який відповідає за розв’язання індивідуальних задач, виводить результат роботи у вигляді діаграми Ґанта, що є прикладом зручного та інформативного користувацького інтерфейсу.

ПЕРЕЛІК ПОСИЛАНЬ

1. Michael L. Pinedo, “The Makespan without Preemptions”. Theory, Algorithms and Systems. Third Edition. New York, NY, USA: Springer, 2008, pp. 106-128.
2. Peter Brucker, “Scheduling Algorithms”. Theory, Algorithms and Systems. Third Edition. Springer Berlin, Heidelberg, Deutschland: Springer Book Archive, 2013, pp. 93-115.
3. Frank Werner, Larysa Burtseva and Yuri Sotskov, “Algorithms for Scheduling Problems”. Theory, Algorithms and Systems. Fourth Edition. Basel, Switzerland: MDPI, 2018, pp. 57-74.
4. Орловський С.О., «Проблеми прийняття рішень за нечіткої вихідної інформації». Навчальний посібник. Перше видання. Київ, Україна: КПІ, 2016, сс. 132-195.
5. Гайна Г.А., «Основи проектування баз даних». Навчальний посібник. Перше видання. Київ, Україна: КНУБА, 2005, сс. 178-206.
6. C# programming guide [Електронний ресурс]: Режим доступу – https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide
7. ASP.NET Core MVC documentation [Електронний ресурс]: Режим доступу – https://docs.microsoft.com/en-us/aspnet/core
8. EF (Entity Framework) documentation [Електронний ресурс]: Режим доступу – https://docs.microsoft.com/en-us/ef/
9. EF (Entity Framework) Core documentation [Електронний ресурс]: Режим доступу – https://docs.microsoft.com/en-us/ef/core
10. MySQL documentation [Електронний ресурс]: Режим доступу – https://dev.mysql.com/doc/
11. AWS documentation [Електронний ресурс]: Режим доступу – https://docs.aws.amazon.com/
12. Route 53 documentation [Електронний ресурс]: Режим доступу – https://docs.aws.amazon.com/route53/index.html

13. Virtual Private Cloud documentation [Електронний ресурс]: Режим доступу – https://docs.aws.amazon.com/vpc/index.html
14. Elastic Compute Cloud documentation [Електронний ресурс]: Режим доступу – https://docs.aws.amazon.com/ec2/index.html
15. Relational Database Service documentation [Електронний ресурс]: Режим доступу – https://docs.aws.amazon.com/rds/index.html
16. Visual Studio documentation [Електронний ресурс]: Режим доступу – https://visualstudio.com/docs
17. Visual Studio Code documentation [Електронний ресурс]: Режим доступу – https://code.visualstudio.com/docs/languages/csharp
18. Git documentation [Електронний ресурс]: Режим доступу – https://git-scm.com/doc