Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет комп'ютерних наук
(повна назва)

Кафедра програмної інженерії
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

Дослідження методів та алгоритмів для побудови оптимізованих розкладів.
(тема)

Виконав:
студент (ка) 2 курсу, групи ІПЗм-22-6

Богацькій Є.Р.
(прізвище, ініціали)

Спеціальність 121 – Інженерія програмного забезпечення
(код і повна назва спеціальності)
Тип програми освітньо-наукова

Керівник проф. Шубін І.Ю.
(посада, прізвище, ініціали)

Допускається до захисту
Зав. кафедри _____ З.В.Дудар
(підпис) (прізвище, ініціали)

2024 р.

Харківський національний університет радіоелектроніки

Факультет комп'ютерних наук

Кафедра програмної інженерії

Рівень вищої освіти другий (магістерський)

Спеціальність 121 – Інженерія програмного забезпечення
(шифр і назва)

Тип програми освітньо-наукова програма

Освітня програма Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«___» _____ 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Богацькому Єгору Романовичу
(прізвище, ім'я, по батькові)

1. Тема роботи «Дослідження методів та алгоритмів для побудови оптимізованих розкладів.»

Затверджена наказом по університету від 29.03. 2024р. № 250 Ст

2. Термін подання студентом роботи до екзаменаційної комісії 25.06.2024

3. Вихідні дані до роботи: Дослідження методів та алгоритмів для побудови оптимізованих розкладів.

4. Перелік питань, що потрібно опрацювати в роботі: Метою роботи є дослідження методів та алгоритмів для побудови оптимізованих розкладів. Основні завдання включають аналіз та порівняння існуючих методів і технік розробки розкладів, доскональне вивчення принципів роботи кожного з обраних підходів, розробка рекомендацій щодо впровадження найбільш ефективних методів, експериментальне створення власного оптимізованого розкладу та проведення тестування і оптимізації розробленого рішення.

КАЛЕНДАРНИЙ ПЛАН

№	Назви етапів курсової роботи	Термін виконання етапів роботи	Примітка
1	Видача теми, узгодження і затвердження	02.04.2024	виконано
2	Аналіз предметної галузі	02.04.2024	виконано
3	Огляд існуючих методів	09.04.2024	виконано
4	Оформлення пояснювальної записки	05.05.2024	виконано
5	Здача готового проекту	25.06.2024	виконано

Дата видачі завдання 20 січня 2024 р.

Студент _____ Богацькій Є.Р.
(підпис) (прізвище, ініціали)

Керівник кваліфікаційної роботи _____ проф. Шубін І.Ю.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ / ABSTRACT

Пояснювальна записка: 65 с., 10 рисунків, 11 джерел.

АЛГОРИТМИ, ІНТЕГРАЦІЯ, МЕТОДИ, ОПТИМІЗАЦІЯ, РОЗКЛАДИ.

Об'єктом глибокого аналізу та дослідження є методи та алгоритми для побудови оптимізованих розкладів у контексті "Intelligent Scheduling Systems". Задачею дослідження є аналіз, розробка та впровадження ефективних методів і технік створення розкладів, які сприяють підвищенню якості та ефективності управління ресурсами.

У процесі наукового дослідження застосовуються передові підходи сучасних технологій, зокрема ті, що пов'язані з оптимізацією розкладів. Великий акцент приділяється розгляду різноманітних алгоритмів і методів, їх адаптації до потреб конкретних завдань, а також інтеграції з існуючими системами управління.

Основні напрямки дослідження включають аналіз існуючих методів побудови розкладів, визначення ключових вимог до оптимізації розкладів та розробку рекомендацій щодо впровадження оптимізованих алгоритмів. Серед досліджуваних методів – лінійне програмування, жадібні алгоритми, генетичні алгоритми, алгоритми CSP та використання штучних нейронних мереж і машинного навчання.

В результаті дослідження розробляються та впроваджуються комплексні стратегії для створення та оптимізації розкладів. Це сприяє не лише покращенню якості та продуктивності управління ресурсами, але й забезпечує успішну інтеграцію змін у динамічному середовищі сучасних технологій. Комбінування різних алгоритмів, зокрема бектрекінгу та генетичних алгоритмів, дозволяє досягати більш високих результатів у пошуку оптимальних рішень, враховуючи різноманітні потреби та обмеження в реальному часі.

ALGORITHMS, INTEGRATION, METHODS, OPTIMIZATION, SCHEDULES.

The object of in-depth analysis and research is the methods and algorithms for constructing optimized schedules in the context of "Intelligent Scheduling Systems". The aim of the research is to analyze, develop, and implement effective methods and techniques for creating schedules that enhance the quality and efficiency of resource management.

In the process of scientific research, advanced approaches in modern technology, particularly those related to schedule optimization, are applied. A significant focus is placed on examining various algorithms and methods, adapting them to the needs of specific tasks, and integrating them with existing management systems.

The main research directions include analyzing existing scheduling methods, determining key requirements for schedule optimization, and developing recommendations for the implementation of optimized algorithms. The methods studied include linear programming, greedy algorithms, genetic algorithms, CSP algorithms, and the use of artificial neural networks and machine learning.

As a result of the research, comprehensive strategies for creating and optimizing schedules are developed and implemented. This not only improves the quality and productivity of resource management but also ensures the successful integration of changes in the dynamic environment of modern technologies. Combining various algorithms, particularly backtracking and genetic algorithms, allows for achieving higher results in finding optimal solutions, considering diverse needs and constraints in real-time.

Я, Богацькій Єгор Романович, студент гр. ІПЗм-22-6, здобувач вищої освіти на другому (магістерському) рівні кафедри «Програмна інженерія», заявляю: моя робота на тему «Дослідження методів та алгоритмів для побудови оптимізованих розкладів», що буде представлена в екзаменаційну комісію для публічного захисту, виконана самостійно, в ній не містяться елементи плагіату і вона може бути опублікована в електронному архіві відкритого доступу ElArKhNURE. Всі запозичення з друкованих та електронних джерел мають відповідні посилання.

Я ознайомлений(на) з діючим положенням «Про протидію академічному плагіату в ХНУРЕ», згідно з яким виявлення плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту та застосування дисциплінарних заходів.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

ІПЗ - Інформаційно-Пошукова Система

СУБД - Система Управління Базами Даних

API - Application Programming Interface

C# - Мова програмування C#

FPGA - Field-Programmable Gate Array (Програмована логічна матриця)

RAM - Random Access Memory (Оперативна пам'ять)

CPU - Central Processing Unit (Центральний процесор)

XML - Extensible Markup Language (Розширювана Мова Розмітки)

JSON - JavaScript Object Notation (Нотація Об'єктів JavaScript)

ВСТУП

У наш час неодмінно зростає кількість оф-лайн закладів різної направленості, та онлайн компанії, для всіх них дуже важливо ефективно використовувати ресурси, починаючи від часу співробітників, закінчуючи розподілом коштів.

Загалом бізнесу необхідно контролювати кожен елемент своєї роботи щоб максимально зменшити витрати та зростити прибутки, не говорячи про те що без деяких елементів бізнес зовсім не зможе функціонувати.

Люди є найбільш складним інструментом у плані формування їх задач, зарплати та часу виділеного на певну задачу. Тому різноманітному бізнесу, не залежно від його направленості необхідні елементи внутрішньої системи які будуть контролювати розподіл ресурсів.

Подібними задачами можуть звісно займатись бухгалтери, системні адміністратори, менеджери проектів та умовні логісти загального розуміння цього слова, але є процеси які не можливо або дуже дорого оптимізувати та підтримувати за рахунок персоналу, тож стає проблема знаходження автоматизованих рішень.

Розклад є саме такою проблемою, адже він відповідає за розподілення часу, людських ресурсів та грошей бізнесу, тому наразі стає доволі важливим створення автоматизованих розкладів типу "Intelligent Scheduling Systems".

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

1.1 Загальний огляд системи розкладу та його формування

Спершу розглянемо загально – що таке розклад, які задачі він вирішує і які моменти його створення потребують доопрацювання та автоматизації.

Розклад – це план або таблиця, що вказує час і послідовність різних дій або подій. Це систематизований спосіб організації часу, який допомагає ефективно управляти різними завданнями чи заходами, що відбуваються протягом дня, тижня, місяця або навіть року. Розклад може бути як дуже детальним, так і більш загальним, в залежності від потреб та контексту його використання.

Розклад вирішує наступні проблеми:

- ефективне управління часом: найважливіша функція розкладу - це допомога в ефективному управлінні часом. він дозволяє розподілити доступний час на конкретні завдання або діяльності, забезпечуючи тим самим більш продуктивне й організоване використання часу;
- зменшення стресу і проблемних ситуацій: розклад допомагає уникнути стресу та плутанини, що виникають, коли люди намагаються впоратися з численними завданнями без ясного плану дій. Чіткий розклад забезпечує структуру і знижує рівень напруженості, пов'язаної з невпевненістю і непередбачуваністю;
- підвищення продуктивності: використання розкладу може значно підвищити продуктивність, оскільки він допомагає зосередитись на завданнях, мінімізувати відволікання і забезпечує ритмічність роботи;
- краще планування та прогнозування: розклад дозволяє краще планувати майбутні події і роботи, а також допомагає у прогнозуванні часу, необхідного для виконання різних завдань;
- підтримка балансу між роботою та особистим життям: розклади можуть допомогти збалансувати професійні і особисті обов'язки, відведення часу як для роботи, так і для відпочинку чи сімейних справ;

- покращення здійснення командної роботи: у професійному середовищі розклади сприяють кращій координації дій команди, дозволяючи членам команди ефективно співпрацювати та узгоджувати свої зусилля;
- підвищення відповідальності та самодисципліни: дотримання розкладу вимагає високого рівня самодисципліни та підвищує відповідальність індивіда або команди за виконання запланованих завдань вчасно;
- поліпшення комунікації та прозорості: у організаційному контексті, чіткий розклад допомагає забезпечити прозорість у комунікації та сприяє більш ефективному розумінню очікувань і дедлайнів серед членів команди;
- адаптація до змін та гнучкість: хоча розклад забезпечує структуру, він також може бути достатньо гнучким, щоб адаптуватися до неочікуваних змін або непередбачуваних обставин;
- покращення якості життя: організований підхід до життя через розклади може покращити загальну якість життя, забезпечуючи достатньо часу для роботи, відпочинку, хобі, та сімейних зобов'язань;
- зниження ризику забування важливих подій або завдань: розклад діє як нагадування про важливі зустрічі, події, дедлайни та інші ключові моменти, що знижує ймовірність пропуску важливих моментів.

У сукупності, розклад є ключовим інструментом управління часом, який вирішує безліч проблем, пов'язаних з організацією, продуктивністю, стресом та загальною якістю життя[1]. Він забезпечує структуру і порядок у повсякденному житті, допомагаючи індивідам і організаціям досягати своїх цілей більш ефективно та з меньшими витратами часу та ресурсів.

"Intelligent Scheduling Systems" (Інтелектуальні системи розкладів) – це передові технологічні рішення, які використовують методи штучного інтелекту (ШІ) та машинного навчання для оптимізації процесу планування і управління часом. Ці системи здатні автоматично аналізувати великі обсяги даних, враховуючи численні змінні та обмеження, для створення більш ефективних і адаптивних розкладів.

Основна перевага інтелектуальних систем розкладів полягає в їх здатності до швидкого реагування на зміни, адаптації до нових умов, і врахування унікальних потреб кожного конкретного випадку. Вони можуть враховувати різноманітні фактори, такі як персональні переваги, ресурсні обмеження, терміни виконання, і навіть прогнозовані зміни в попиті або ринкових умовах.

Ці системи широко застосовуються в різних галузях, включаючи виробництво, логістику, охорону здоров'я, освіту, і корпоративне управління. Вони допомагають підприємствам та організаціям підвищувати ефективність, знижувати витрати та покращувати задоволеність клієнтів або співробітників.

Завдяки інтеграції з іншими технологіями, як-от Інтернет речей (IoT) та великі дані (Big Data), інтелектуальні системи розкладів стають ще більш потужними. Вони здатні аналізувати реальні дані у режимі реального часу для створення адаптивних та гнучких розкладів, що відповідають динамічним умовам бізнесу та особистого життя.

Також варто розглянути створення розкладу в закладах обслуговування з урахуванням побажань працівників. Тож коли розклад враховує особисті переваги та потреби працівників, це підвищує їх задоволеність роботою. Задоволені співробітники зазвичай більш мотивовані та продуктивні, що позитивно впливає на загальну ефективність роботи закладу.

Також гнучкий розклад, який враховує індивідуальні обставини, може допомогти співробітникам краще управляти балансом між роботою та особистим життям, знижуючи ризик професійного вигорання.

Працівники, які відчувають, що їхні потреби та бажання враховуються при складанні розкладу, менш схильні шукати роботу в іншому місці. Таким чином, компанія може знизити плинність кадрів та витрати на набір та навчання нових співробітників.

Це дозволить працівникам мати комфортний робочий план та гарну мотивацію. Мотивовані та задоволені співробітники часто надають краще обслуговування клієнтам. Це підвищує рівень задоволеності клієнтів та сприяє позитивному іміджу компанії.

Урахування побажань співробітників може зробити розклад більш гнучким і адаптивним до змін, таких як несподівані піки або спади у попиті на послуги.

Це сприяє створенню позитивної робочої культури, показуючи турботу про благополуччя та вподобання своїх співробітників, компанії сприяють створенню позитивної, взаємо поважної робочої культури. Це, в свою чергу, може залучити талановитих працівників та підвищити загальну репутацію компанії на ринку.

Врахування побажань співробітників при складанні розкладу не тільки підвищує їхнє задоволення та продуктивність, але й сприяє загальному успіху та стабільності компанії. Це створює ситуацію, у якій виграють усі сторони.

1.2 Актуальність проблеми

У сучасному світі, де час і ресурси є ключовими факторами успіху в будь-якому бізнесі, актуальність створення автоматизованих розкладів набуває особливої ваги.

Використання автоматизованих систем для створення розкладів не тільки підвищує ефективність використання ресурсів, але й відіграє вирішальну роль у збалансуванні робочих навантажень та задоволенні потреб персоналу.

Автоматизація розкладів зводить до мінімуму людські помилки та знижує час, необхідний для їх розробки. Традиційний підхід до складання розкладів вручну часто є часозатратним та схильним до помилок, оскільки вимагає врахування великої кількості факторів і варіабельності. Автоматизовані системи системи зазвичай прямолінійні та не враховують багатьох факторів впливу.

Також варто відмітити що авизація розкладів забезпечує більшу справедливість та задоволення працівників. Системи можуть враховувати індивідуальні побажання персоналу, їхні кваліфікації та доступність, забезпечуючи рівномірний розподіл навантаження.

Це підвищує мотивацію працівників і знижує ризик вигоряння, оскільки робота розподіляється більш справедливо.

Збалансований розклад, що враховує як потреби працівників, так і вимоги закладу, сприяє підвищенню продуктивності та ефективності роботи.

Автоматизовані системи можуть адаптуватися до змін у попиті та оперативно коригувати розклади відповідно до поточних потреб закладу.

Основною проблемою у великих організаціях – є те що робітників багато і у сіх робітників відрізняються бажання що до особистого розкладу, це характерно для закладів харчування: ресторанів, кафе і так далі., отже стає проблема того – скільки людей та якої кваліфікації потрібно для вирішення поставлених закладом або ситуацією задач.

Саме тому необхідно розробити алгоритм який дозволить вирішити усі поставлені задачі та врахувати багато факторів впливу, а також бути гнучким до змін.

1.3 Постановка задачі

Уході даної роботи нами буде досліджено предметну область застосування алгоритмів які зможуть вирішити проблему розробки та оптимізації розкладу типу "Intelligent Scheduling Systems", також буде досліджено предметну область розкладів, та проблем які із ними пов’язані.

Ми проведемо порівняльний аналіз кожного із методів, порівняємо з огляду на формування розкладу, далі проведемо поглиблений аналіз адаптованої функціональності відносно наших задач.

Ця робота спрямована на виявлення найкращого методу, алгоритму або модифікованого алгоритму який буде вирішувати проблему створення та оптимізації розкладу який буде задовольняти умовам робітників та закладу чи компанії.

Реалізація наукового дослідження складається з наступних етапів:

– аналіз предметної області;

– аналіз архітектурних особливостей алгоритмів для оптимізації розкладів;

– розгляд їх основних особливостей;

– порівняльний аналіз та виведення підсумків за вибором;

– модернізація та створення нового алгоритму.

2 АНАЛІЗ ПЕРСПЕКТИВНИХ МЕТОДІВ ТА АЛГОРИТМІВ

2.1 Лінійне програмування

Лінійне програмування – це математичний метод для оптимізації лінійної цільової функції, заданої з використанням лінійних рівнянь та нерівностей.

Цей метод широко використовується для вирішення різних задач у таких сферах, як бізнес, економіка, інженерія, військові плани та інші області, де потрібно максимізувати або мінімізувати лінійну функцію під набором лінійних обмежень.

Лінійне програмування дозволяє ефективно знаходити оптимальні рішення в умовах, де обмеження та цілі можуть бути точно виражені лінійними рівняннями.

Подібний підхід у контексті оптимізації розподілу ресурсів використовується для максимізації або мінімізації лінійної цільової функції (наприклад, прибутку, витрат, ефективності виробництва) за умови дотримання лінійних обмежень (наприклад, обмежені ресурси, трудові норми).

Використання лінійного програмування дозволяє ефективно розподілити обмежені ресурси (такі як сировина, час, фінанси) для досягнення оптимального результату, який задовольняє всі встановлені обмеження.

Цей метод широко застосовується в різних областях для вирішення практичних задач управління та планування.

Подібний підхід для оптимізації розподілу ресурсів включає більш складні концепції, які можна розглянути детальніше:

- цільова функція – лінійна функція, яка виражає мету, що потрібно максимізувати або мінімізувати (наприклад, витрати, прибуток, виробництво);
- змінні рішення – представляють кількісні аспекти рішення, такі як кількість продуктів для виробництва або години роботи;

- обмеження – лінійні рівняння або нерівності, що встановлюють умови, яким мають задовольняти змінні (наприклад, обмежена кількість ресурсів або часу);
- оптимізаційний процес – шукає комбінацію змінних, яка найкраще задовольняє цільову функцію в межах обмежень;
- симплекс-метод – один з алгоритмів для розв'язання завдань лінійного програмування, який систематично перевіряє кути обмежувального багатогранника рішень для знаходження оптимального рішення;
- геометрична інтерпретація – обмеження утворюють багатогранник у просторі рішень, а оптимальне рішення зазвичай знаходиться на одній з його вершин.

Отже розглянутий алгоритм є потужним інструментом для оптимізації, дозволяючи приймати обґрунтовані рішення в умовах обмежених ресурсів.

Далі на рисунку 2.1 наведено взаємодію елементів цього підходу, його компоненти та зв’язок між ними[2].

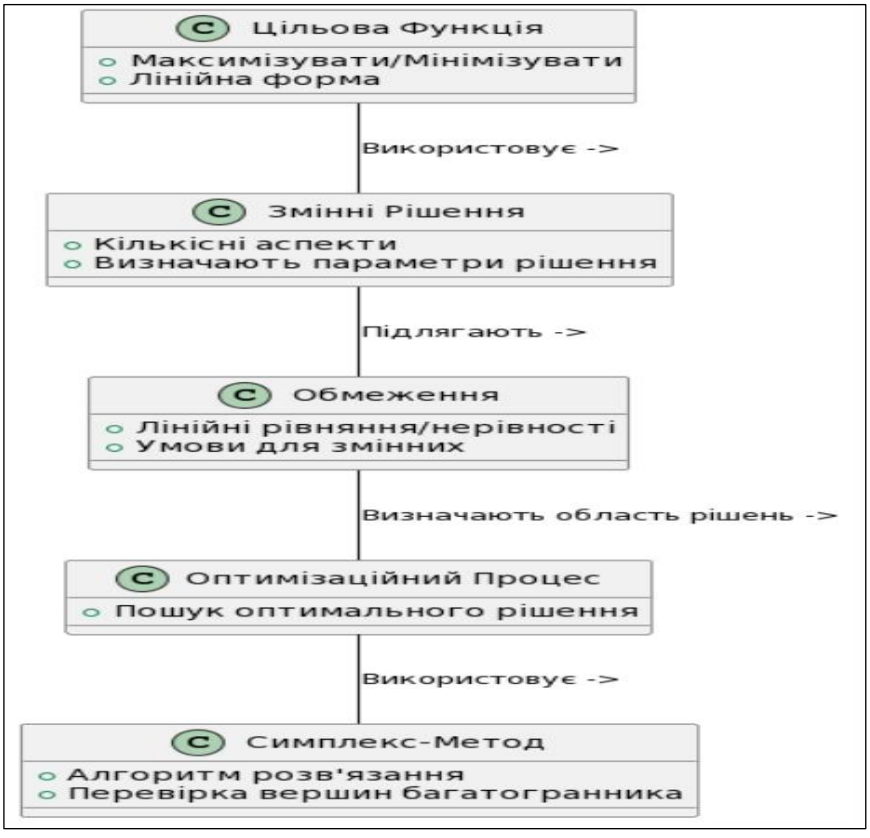

Рисунок 2.1 – Взаємодія складових елементів лінійного програмування.

Цільова Функція визначає кінцеву мету оптимізації (наприклад, максимізацію прибутку або мінімізацію витрат). Вона формулюється через лінійні вирази, які включають "Змінні Рішення".

Змінні Рішення є кількісними параметрами, над якими проводиться оптимізація. Вони впливають на результат цільової функції і обмежені через "Обмеження".

Обмеження ставлять межі для можливих значень "Змінних Рішення", обмежуючи простір допустимих рішень. Ці обмеження можуть включати ресурсні ліміти, часові обмеження тощо.

Оптимізаційний Процес полягає в пошуку такого набору значень "Змінних Рішення", які задовольняють усі "Обмеження" та оптимізують "Цільову Функцію".

Симплекс-Метод (або інший алгоритм) використовується в "Оптимізаційному Процесі" для ефективного знаходження оптимального рішення, аналізуючи вершини багатогранника рішень, утвореного обмеженнями.

Таким чином, ці компоненти створюють інтегровану систему для ефективного розподілу ресурсів.

Лінійне програмування може бути адаптовано для створення розумного розкладу, оскільки дозволяє оптимізувати розподіл обмежених ресурсів (наприклад, часу, персоналу) при дотриманні різних обмежень. У контексті розкладу, цільова функція може включати максимізацію використання ресурсів або мінімізацію затрат часу.

Змінні рішення представляють розподіл часу чи ресурсів. Обмеження можуть включати часові рамки, доступність персоналу, обмеження приміщень тощо. Використання лінійного програмування дозволяє знайти найоптимальніший розклад, що задовольняє всім обмеженням та максимізує ефективність розподілу ресурсів.

Приклад використання лінійного програмування для створення та оптимізації розкладу:

– цільова функція: мінімізувати загальний час простою персоналу;

– змінні рішення: кількість годин, призначених кожному працівнику;
– обмеження: кожен працівник має працювати не більше 40 годин на тиждень; певні зміни вимагають мінімальної кількості персоналу; кімнати доступні лише у певні години.

Отже підхід дозволяє розрахувати оптимальний розподіл робочих годин для кожного працівника, максимізуючи ефективність робочого процесу і зменшуючи неефективний час простою, враховуючи всі обмеження.

2.2 Алгоритм жадібної оптимізації (Algorithm of greedy optimization)

Алгоритми жадібної оптимізації – це клас алгоритмів, що вирішують оптимізаційні задачі шляхом прийняття локально оптимальних рішень на кожному кроці з метою знайти глобально оптимальне рішення. Іншими словами, на кожному кроці алгоритм вибирає найкращий варіант з усіх доступних, не зважаючи на більш далекосяжні наслідки.

Жадібні алгоритми ефективні та часто прості у реалізації, але не завжди гарантують знаходження найкращого можливого рішення для всіх задач.

Алгоритми жадібної оптимізації у контексті оптимізації розподілу ресурсів використовують простий підхід: на кожному етапі вони вибирають найкраще локальне рішення з надією досягти глобально оптимального результату[3].

Наприклад, при розподілі робочих годин серед працівників алгоритм може спробувати мінімізувати загальний час простою, вибираючи співробітників з найнижчими витратами на робочу годину на кожному кроці.

Жадібні алгоритми є простими у реалізації та ефективними у випадках, де локально оптимальні рішення часто ведуть до глобально оптимального результату.

Для детального розуміння алгоритмів жадібної оптимізації у контексті розподілу ресурсів варто розглянути наступні ключові елементи та механізми:

– принцип жадібності – центральна ідея полягає у виборі найкращого доступного рішення на кожному кроці, без спроби передбачити майбутні наслідки цього вибору;

– локальна оптимізація – жадібні алгоритми приймають рішення, які виглядають найкращими в короткостроковій перспективі. наприклад, призначення ресурсів на основі поточної найбільшої потреби;
– ітеративний процес – алгоритм продовжує вибирати локально оптимальні рішення на кожному кроці до досягнення кінцевої мети або вичерпання ресурсів;
– простота реалізації – жадібні алгоритми зазвичай простіше програмувати і швидші за інші методи оптимізації;
– обмеження – жадібні алгоритми можуть не завжди знаходити глобально оптимальне рішення, особливо в складних або багатокритеріальних задачах.

Далі на рисунку 2.2 наведено взаємодію елементів цього підходу, його компоненти та зв’язок між ними.

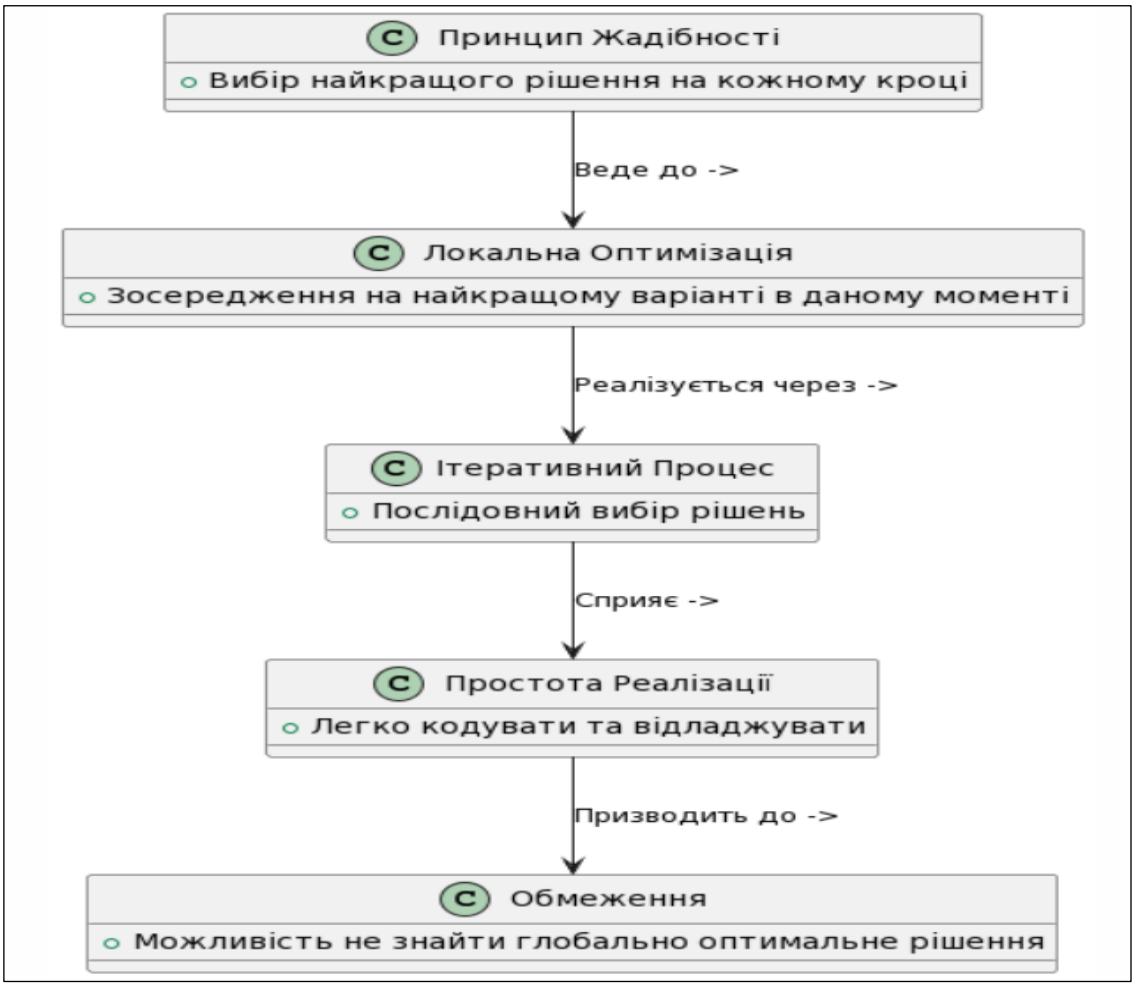

Рисунок 2.2 – Взаємодія складових елементів алгоритми жадібної оптимізації.

Жадібні алгоритми можуть бути адаптовані для створення розумного розкладу наступним чином:

- визначення пріоритетів: встановіть критерії, за якими буде вибиратися кожне рішення (наприклад, заповнення найбільш вимогливих змін спочатку);
- послідовний вибір: на кожному кроці вибирайте опцію, яка найкраще відповідає критеріям, наприклад, призначення співробітників з найвищою кваліфікацією на найважливіші задачі;
- адаптація до змін: жадібні алгоритми можуть швидко адаптуватися до змін, оскільки вони не вимагають повторного розгляду всього розкладу при зміні одного аспекту.

Такий підхід дозволяє швидко створити розклад, зосереджуючись на найважливіших елементах спочатку і забезпечуючи ефективне використання ресурсів. Однак важливо враховувати, що цей підхід може не завжди знаходити найкраще можливе рішення для всього розкладу.

Далі наведемо приклад використання жадібного алгоритму для створення та оптимізації розкладу.

Уявімо, що потрібно скласти розклад для викладачів у школі, максимізуючи ефективність використання класних кімнат та викладацького часу. Реалізація включає наступні кроки:

- визначення пріоритетів – спочатку визначаємо пріоритетні предмети або класи, які потребують особливої уваги або мають жорсткі часові рамки;
- жадібний вибір – на кожному кроці алгоритм вибирає викладача з найбільшим досвідом для пріоритетного класу або предмету;
- розподіл ресурсів – після призначення викладачів для пріоритетних класів, алгоритм переходить до наступних класів, використовуючи залишкові ресурси (вільних викладачів і класи).

Цей підхід дозволяє швидко скласти розклад, зосередившись на найважливіших аспектах спочатку, але може не враховувати більш складні взаємозв'язки або потреби, які виявляються на більш пізніх етапах.

2.3 Генетичні алгоритми

Генетичні алгоритми – це методи оптимізації та пошуку рішень, які імітують процеси природного відбору та генетики. Вони використовують поняття хромосом (репрезентація рішень), мутації, схрещування та відбору для генерації нових поколінь рішень. Генетичні алгоритми часто застосовуються для розв'язання складних оптимізаційних задач, де звичайні методи неефективні.

Вони ефективні в задачах з великим простором пошуку, де потребують вибору найкращого рішення з багатьох можливих варіантів. Генетичні алгоритми особливо корисні, коли прямий пошук кожного можливого варіанту непрактичний через їхню велику кількість.

Також використовують принципи біологічної еволюції, такі як спадковість, мутація, селекція та схрещування, для поступового вдосконалення рішень. В результаті, генетичні алгоритми можуть знаходити ефективні рішення для складних проблем, адаптуючись та оптимізуючи в процесі пошуку.

Генетичні алгоритми можуть бути застосовані для оптимізації розкладів, використовуючи принципи природного відбору та генетичної мутації. У цьому контексті, різні можливі розклади можуть бути представлені як окремі "індивіди" в популяції. Ці індивіди "схрещуються" та "мутують" в процесі, що створює нові варіанти розкладів[4].

Найкращі розклади обираються на основі визначених критеріїв (наприклад, мінімізація часу простою, забезпечення рівномірного розподілу навантаження). Цей процес повторюється протягом багатьох "поколінь", поки не буде знайдено оптимальне рішення або поки не будуть вичерпані ресурси обчислень.

Далі наведемо приблизний сценарій використання цього алгоритму для створення розкладу, він має наступні етапи:

– ініціалізація популяції: на цьому етапі створюється велика кількість різноманітних розкладів, які слугують початковими кандидатами для оптимізації. кожен розклад представлений як набір параметрів, що кодується в формі «хромосоми»;

– оцінка приспособленості: кожен розклад оцінюється за допомогою спеціально розробленої функції приспособленості, яка вимірює, наскільки добре розклад задовольняє встановлені критерії (наприклад, ефективність розподілу часу);
– селекція: вибираються найкращі розклади з поточної популяції. це може бути здійснено за допомогою різних методів, таких як турнірна селекція або рулеткова селекція;
– схрещування (кросовер): пари вибраних розкладів комбінуються для створення нових розкладів. цей процес включає обмін сегментами між хромосомами батьківських розкладів;
– мутація: для підтримки різноманітності в популяції вносяться невеликі випадкові зміни в хромосоми розкладів;
– нове покоління: створене нове покоління розкладів замінює попереднє. цей процес продовжується до тих пір, поки не буде знайдено задовільний розклад або досягнуто заданого числа ітерацій.

Далі на рисунку 2.3 наведено взаємодію елементів цього підходу, його компоненти та зв’язок між ними.

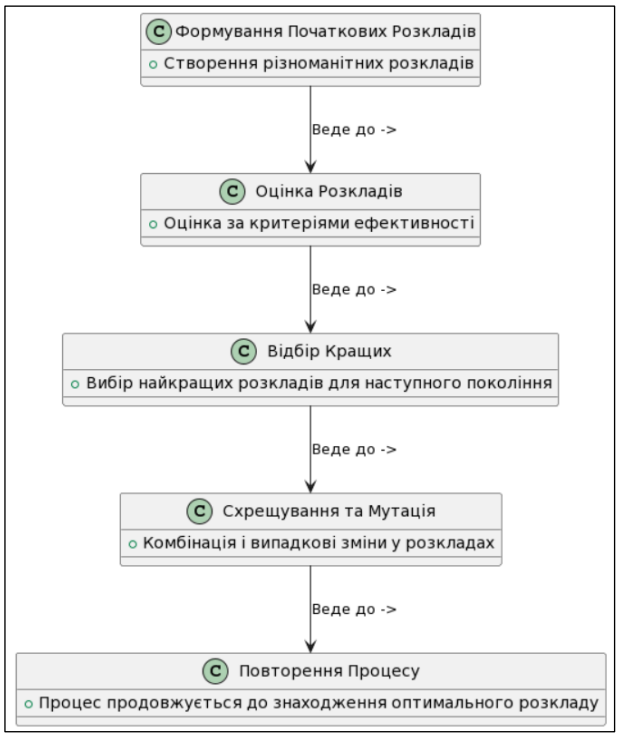

Рисунок 2.3 – Взаємодія складових елементів генетичного алгоритму

Наведена модель представляє взаємозв'язок між основними елементами генетичного алгоритму для оптимізації розкладу, починаючи від формування початкових розкладів, оцінки цих розкладів, відбору кращих, до схрещування та мутації, і, нарешті, до повторення процесу до досягнення оптимального розкладу.

Конфлікти між схрещеними розкладами також будуть вирішуватись через пріоритетність конфліктуючих елементів.

2.4 Алгоритми Constraint Satisfaction Problem (CSP)

Алгоритми Constraint Satisfaction Problem (CSP) використовуються для розв'язання задач, де потрібно знайти значення для набору змінних, що задовольняють певні обмеження. Ключовими елементами CSP є змінні, їхні можливі значення (домени) та обмеження, які визначають допустимі комбінації значень цих змінних.

Цей алгоритми CSP шукають рішення, де всі обмеження виконуються, і часто використовуються у плануванні, розкладах, аналізі та інших областях, де існують складні взаємозв'язки або обмеження.

У проблемі задоволення обмежень є в основному три основні компоненти:

- змінні: те, що потрібно визначити, є змінними. змінні в csp – це об’єкти, яким повинні бути призначені значення, щоб задовольнити певний набір обмежень. логічні, цілі чи категоричні змінні – це лише кілька прикладів різних типів змінних. змінні, наприклад, можуть замінити багато клітинок головоломки, які потрібно заповнити числами в головоломці судоку.
- домени: діапазон потенційних значень, які може мати змінна, представлений доменами. залежно від проблеми домен може бути обмеженим або необмеженим. наприклад, у судоку набір чисел від 1 до 9 може служити доменом змінної, що представляє проблемну комірку.
- обмеження: вказівки, які контролюють, як змінні співвідносяться одна з одною, відомі як обмеження. обмеження в csp визначають діапазони можливих значень для змінних. унарні обмеження, двійкові обмеження

та обмеження вищого порядку є лише кількома прикладами різноманітних видів обмежень. наприклад, у задачі судоку обмеження можуть полягати в тому, що кожен рядок, стовпець і квадрат 3×3 можуть мати лише один екземпляр кожного числа від 1 до 9.

Далі розглянемо приклад відображення проблеми задоволення встановлених обмежень.

Приклад проблем задоволення обмежень (CSP):

- кінцевий набір змінних v1, v2, в3 ……..вn;
- непустий домен кожної окремої змінної d1, d2< a i=4>, d3……..dn;
- кінцевий набір обмежень c1, c2 ……, див., де кожне обмеження сі обмежує можливі значення змінних, наприклад, v1 ≠ v2;
- кожне обмеження сі є парою <область дії, відношення>;
- приклад: <(v1, v2), v1 не дорівнює v2>;
- область дії = набір змінних, які у обмеженні;
- відношення = перелік допустимих комбінацій значень змінних.

Це може бути чіткий перелік дозволених комбінацій. можливо, це абстрактне відношення, що дозволяє перевіряти членство та складати списки.

Алгоритми Constraint Satisfaction Problem (CSP) працюють за допомогою різних методів для знаходження рішення, яке задовольняє всім обмеженням. Основні методи включають:

- backtracking: – це алгоритм пошуку в глибину, який методично досліджує простір пошуку потенційних рішень, доки не буде виявлено рішення, яке задовольняє всі обмеження. метод починається з вибору змінної та надання їй значення перед повторними спробами надати значення іншим змінним. метод повертається до попередньої змінної та пробує інше значення, якщо в будь-який час змінній не можна надати значення, яке відповідає вимогам. коли всі присвоєння випробувано або знайдено рішення, яке задовольняє всі обмеження, алгоритм завершується;

- forward checking: після призначення значення змінній відразу перевіряються наступні змінні, щоб виключити значення, які порушують обмеження. це допомагає зменшити кількість майбутніх порушень і зводить до мінімуму необхідність відступу;
- arc consistency: перед тим, як призначати значення змінній, алгоритм переконується, що кожна змінна взаємосумісна з іншими в контексті обмежень. це забезпечує, що для кожної змінної існують можливі значення, які не призведуть до порушення обмежень.

CSP використовуються у різноманітних застосуваннях, від розкладів до задач проектування та планування. Вони допомагають управляти складними обмеженнями і знаходити рішення, які задовольняють усі критерії задачі.

Алгоритми CSP можуть бути адаптовані для створення розумного розкладу шляхом моделювання розкладу як набору обмежень. Кожне обмеження визначає умови, які повинні бути виконані.

Розглянемо як приклад обмеження до формування розкладу персоналу ресторану:

- кожен працівник має визначену кількість годин, які він може працювати на тиждень;
- деякі зміни, наприклад, обідні або вечірні, можуть вимагати більшої кількості персоналу;
- працівники мають різні навички (наприклад, бармени, офіціанти, кухарі), і їх розподіл має враховувати ці спеціалізації.
- необхідно враховувати заплановані відпустки та вільні дні працівників.

З обмеженнями, визначеними для розкладу персоналу у ресторані, можна діяти за наступним сценарієм:

- визначити кожне обмеження як частину математичної моделі csp. це включає встановлення змінних для кожного працівника та їх можливих змін;
- використати алгоритм csp, який може враховувати всі ці обмеження. він здійснюватиме пошук рішень, які задовольняють всім обмеженням;

- налаштувати алгоритм так, щоб він не тільки задовольняв обмеженням, але й оптимізував певні цілі, наприклад, максимальне використання персоналу або збалансування робочого навантаження.
- використати отриманий розклад і бути готовим до його коригування у випадку непередбачених змін, таких як хвороби працівників або зміна попиту.

2.5 Штучні нейронні мережі і машинне навчання

Штучний інтелект (ШІ) – це галузь комп'ютерних наук, яка зосереджена на створенні систем, здатних виконувати завдання, які зазвичай вимагають людського інтелекту. Це включає розуміння мови, навчання, планування, розпізнавання образів та ін. ШІ включає різноманітні підходи та технології, наприклад, нейронні мережі та алгоритми машинного навчання.

Машинне навчання, підгалузь ШІ, це методика, за допомогою якої комп'ютерні системи автоматично вчяться та вдосконалюються з досвіду без явного програмування.

Використовуючи алгоритми, які постійно збирають дані, машинне навчання дозволяє системам знаходити приховані узори та передбачати майбутні тренди. Воно включає різні методи, як-от навчання під наглядом, навчання без нагляду та навчання з підкріпленням[5].

ШІ та машинне навчання мають широкий спектр застосувань у різних сферах, від розпізнавання мови та образів до прогнозування фінансових тенденцій та автоматизованого прийняття рішень у бізнесі. Вони відіграють важливу роль у формуванні сучасних технологій і продовжують еволюціонувати, вносячи значні зміни у спосіб, яким ми взаємодіємо з технологіями.

Штучний інтелект (ШІ) та машинне навчання революціонізують створення розкладу за допомогою "Intelligent Scheduling Systems". Використання алгоритмів ШІ дозволяє автоматизувати процес складання розкладів, що значно підвищує їх ефективність та оптимальність. Ці системи можуть аналізувати великі обсяги даних, враховувати складні змінні та обмеження, а також прогнозувати потенційні

проблеми. Вони адаптуються до змінних умов та потреб, забезпечуючи гнучкість та швидкість адаптації. Таким чином, ШІ та машинне навчання дозволяють створювати більш точні та ефективні розклади, мінімізуючи простої та покращуючи загальну продуктивність.

Нейронні мережі та машинне навчання можуть бути ефективно задіяні для вирішення задач розкладу. Це області штучного інтелекту, які імітують спосіб, яким людський мозок аналізує та обробляє інформацію, і є особливо корисними для розв'язання складних проблем, де традиційні алгоритми можуть бути недостатньо ефективними.

Реалізація оптимізації розкладу проходиться у декілька етапів:

- збір та обробка даних. першим кроком є збір історичних даних про розклади. це може включати інформацію про робочі години, переваги персоналу, вимоги до виробництва тощо;
- аналіз даних та виявлення закономірностей. за допомогою технік машинного навчання, таких як класифікація, регресія чи кластеризація, можна виявити закономірності у цих даних. це допомагає визначити, які фактори найбільше впливають на успішний розклад;
- побудова моделі прогнозування. використовуючи виявлені закономірності, можна побудувати моделі, які прогнозують ефективність різних варіантів розкладу;
- оптимізація розкладу з використанням нейронних мереж. нейронні мережі можуть допомогти в оптимізації розкладу, автоматично адаптуючись до змін у вхідних даних і забезпечуючи гнучкість у прийнятті рішень;
- тестування та валідація. отримані моделі потребують тестування та валідації для забезпечення їхньої точності та надійності;
- інтеграція з існуючими системами. налаштування та інтеграція моделей з існуючими системами планування та управління розкладами.

Тепер у контексті нашого розкладу розглянемо покроково кожен етап проходження потенційного алгоритму створення розкладу.

У контексті "Intelligent Scheduling Systems", процес збору та обробки даних відіграє ключову роль:

- збір даних. збираються дані про доступність працівників, попит на роботу, історичні розклади та будь-які особливі обмеження (наприклад, відпустки співробітників);
- первинна обробка. включає очищення даних від помилок, видалення невідповідностей, перевірку на викиди та стандартизацію форматів;
- аналіз даних: застосування статистичних методів та аналітичних інструментів для виявлення закономірностей та визначення ключових факторів, що впливають на ефективність розкладу.

Інтеграція та узагальнення даних у контексті "Intelligent Scheduling Systems" є важливим кроком, що дозволяє системам адекватно обробляти і використовувати інформацію з різних джерел.

Цей процес включає злиття даних з різноманітних систем (наприклад, відділу кадрів, фінансів, логістики) для створення єдиної бази даних. Потім ці дані узагальнюються, щоб виявити основні тенденції та закономірності, які можуть бути використані для оптимізації розкладів. Такий підхід дозволяє системам штучного інтелекту забезпечувати точне і гнучке планування, враховуючи всі аспекти роботи організації.

Аналіз даних та виявлення закономірностей у контексті "Intelligent Scheduling Systems" включає детальний розгляд зібраних даних для ідентифікації шаблонів, що можуть вплинути на ефективність розкладів. Використовуються алгоритми машинного навчання та статистичний аналіз для виявлення кореляцій та залежностей, таких як часові тренди, оптимальні комбінації персоналу, вплив певних подій на робочі навантаження.

Цей аналіз допомагає у передбаченні потреб у персоналі та розробці більш гнучких і адаптивних розкладів.

Підготовка даних для моделювання у контексті "Intelligent Scheduling Systems" – це процес перетворення зібраних даних у формат, який може бути використаний алгоритмами машинного навчання. Він включає нормалізацію

даних для забезпечення консистентності, розбиття даних на тренувальні та тестові набори, кодування категоріальних даних та обробку пропущених значень.

Цей крок важливий для забезпечення точності та надійності майбутніх прогнозів та аналізів, проведених системою.

3 ПОРІВНЯННЯ РІШЕНЬ ТА ЇХ МОДИФІКАЦІЯ

3.1 Порівняння рішень відносно розкладу типу "Intelligent Scheduling Systems"

Далі для вибору найбільш підходящого та релевантного алгоритму – розглянемо недоліки кожного із них, та визначимо які із них найбільш життєздатні.

У контексті розробки розкладу типу "Intelligent Scheduling Systems", використання лінійного програмування може мати декілька мінусів:

- обмеженість у вирішенні тільки лінійних проблем: лінійне програмування ефективно для задач, які можуть бути виражені лінійними рівняннями та обмеженнями. у багатьох реальних сценаріях розкладу, особливо в складних системах, проблеми часто нелінійні;
- складність моделювання динамічних умов: лінійне програмування не завжди ефективно моделює динамічні зміни в умовах, таких як раптові зміни в доступності персоналу або змінені попити;
- високі вимоги до точності даних: для ефективного вирішення, лінійне програмування вимагає дуже точних вхідних даних. в реальному світі така точність даних часто важко досяжна;
- обмеження у врахуванні множинності цілей: лінійне програмування може бути обмежене при врахуванні множинних цілей або пріоритетів, що часто зустрічаються у розкладах.

Таким чином, хоча лінійне програмування може бути корисним для деяких аспектів розробки розкладу, воно може не бути ідеальним рішенням для більш складних або динамічних систем планування.

Жадібні алгоритми можуть мати деякі обмеження при використанні в контексті розробки розкладів типу "Intelligent Scheduling Systems":

- локальні оптимуми: основний недолік жадібних алгоритмів полягає у їхній схильності до застрягання в локальних оптимумах, а не знаходження глобально оптимального рішення;

- неадаптивність: жадібні алгоритми не адаптуються до змін у даних або умовах, оскільки вони не переглядають раніше прийняті рішення;
- складність у прогнозуванні багато-ітераційного розкладу: вони не завжди можуть передбачити довгострокові наслідки прийнятих рішень, оскільки фокусуються на досягненні найкращого короткострокового результату.

Таким чином, хоча жадібні алгоритми можуть бути корисними для певних аспектів планування розкладу, їх обмеження роблять їх не завжди ідеальними для комплексних або динамічних систем.

Далі розглянемо генетичні алгоритми, вони хоча і потужні в оптимізації розкладів мають деякі обмеження:

- часові затрати на обчислення: оптимізація за допомогою генетичних алгоритмів може бути часо затратною, особливо для великих і складних розкладів;
- випадковість та непередбачуваність: через випадковий характер мутацій та схрещувань, іноді важко забезпечити послідовність та передбачуваність результатів поєднання декількох розкладів із своїми обмеженнями;
- ризик застрягання в локальних максимумах: існує ризик, що алгоритм зосередиться на локально оптимальному рішенні, не знайшовши глобального оптимуму.

Алгоритми Constraint Satisfaction Problem (CSP) загалом нам доволі добре підходять, але мають деякі обмеження:

- обмеженість у рішенні тільки чітко визначених обмежень: csp ефективні тільки для варіантів моделі розкладу, де всі обмеження можна чітко визначити та представити у формі конкретних правил;
- складність з великою кількістю змінних: у системах із великою кількістю змінних і складними обмеженнями, розв'язання csp може стати надмірно складним і часо затратним;

– відсутність гнучкості: csp можуть не бути достатньо гнучкими для врахування неочікуваних змін або динамічних умов, які часто виникають у практичних умовах планування розкладу.

Але останній недолік нівелюється поєднанням із алгоритмом мутації який принесе більше гнучкості та зміни стану до отримання оптимального результату змінюючи сети обходу для цього алгоритму.

Але як з’ясовано – хоча CSP є потужним інструментом для вирішення певних задач планування, їх обмеження можуть ускладнити використання в більш динамічних і складних системах планування.

Штучні нейронні мережі та машинне навчання також мають деякі мінуси у контексті розробки розкладу типу "Intelligent Scheduling Systems":

– складність розуміння та інтерпретації: результати, отримані від нейронних мереж, можуть бути складними для інтерпретації, що ускладнює зрозуміння логіки підбору та коригування розкладів;
– залежність від даних: якість та кількість даних, які використовуються для тренування, мають вирішальне значення. недостатньо представлені або зашумлені дані можуть призвести до неправильного навчання моделей;
– ризик перенавчання: існує ризик перенавчання моделей на конкретні дані, що може зменшити їх здатність адекватно реагувати на нові умови або зміни в даних.

3.2 Вибір ключового функціоналу

Алгоритм бектрекінгу у контексті "Intelligent Scheduling Systems" ефективно використовується для складання складних розкладів, де потрібно враховувати безліч змінних і обмежень. Цей алгоритм базується на методі "спробуй і відхили", де послідовно розглядаються всі можливі комбінації розкладу до знаходження відповідного рішення.

У процесі складання розкладу, алгоритм бектрекінгу починає з першого рішення або вибору і рухається вперед, поки не зустріне обмеження або не

досягне кінця можливих рішень. Якщо обмеження зустрічається, алгоритм повертається назад (бектрекінг) і пробує інші варіанти.

Це особливо корисно в ситуаціях, де потрібно балансувати між багатьма змінними, такими як побажання працівників щодо графіку, їх кваліфікація, правила відпусток, пікові години роботи тощо. Бектрекінг дозволяє розглядати кожну можливу комбінацію цих змінних, щоб знайти найоптимальніший розклад, який враховує всі ці фактори.

Проте, слід зазначити, що алгоритм бектрекінгу може бути часозатратним у великих і складних системах, оскільки він перевіряє багато можливих комбінацій. Отже, його ефективність значною мірою залежить від обсягу даних та складності задачі планування. У деяких випадках може бути корисним комбінувати бектрекінг з іншими методами оптимізації, такими як генетичні алгоритми або машинне навчання, для підвищення ефективності процесу складання розкладу.

На рисунку 3.1 наведено схему роботи цього алгоритму, на рисунку продемонстровано гілки по яким проходить за певних умов рішення.

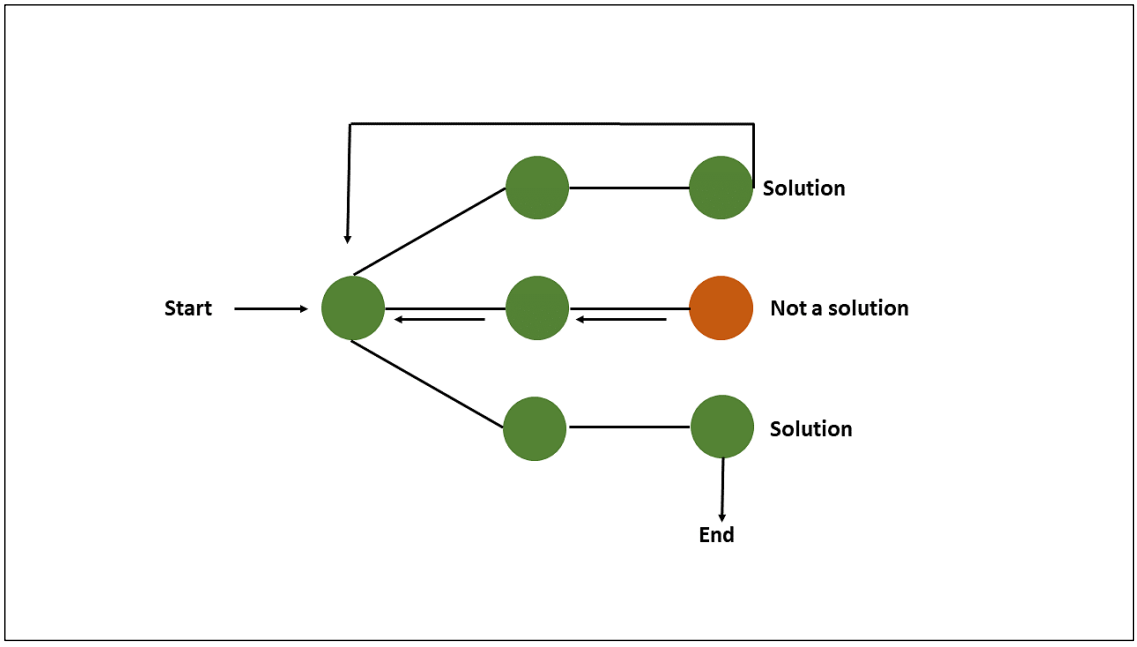

Рисунок 3.1 – Схема роботи алгоритму бактрекінгу

На рисунку 3.2 наведено більш прикладну схему з описом потенційного сценарію його використання у системах оптимізації розкладів.

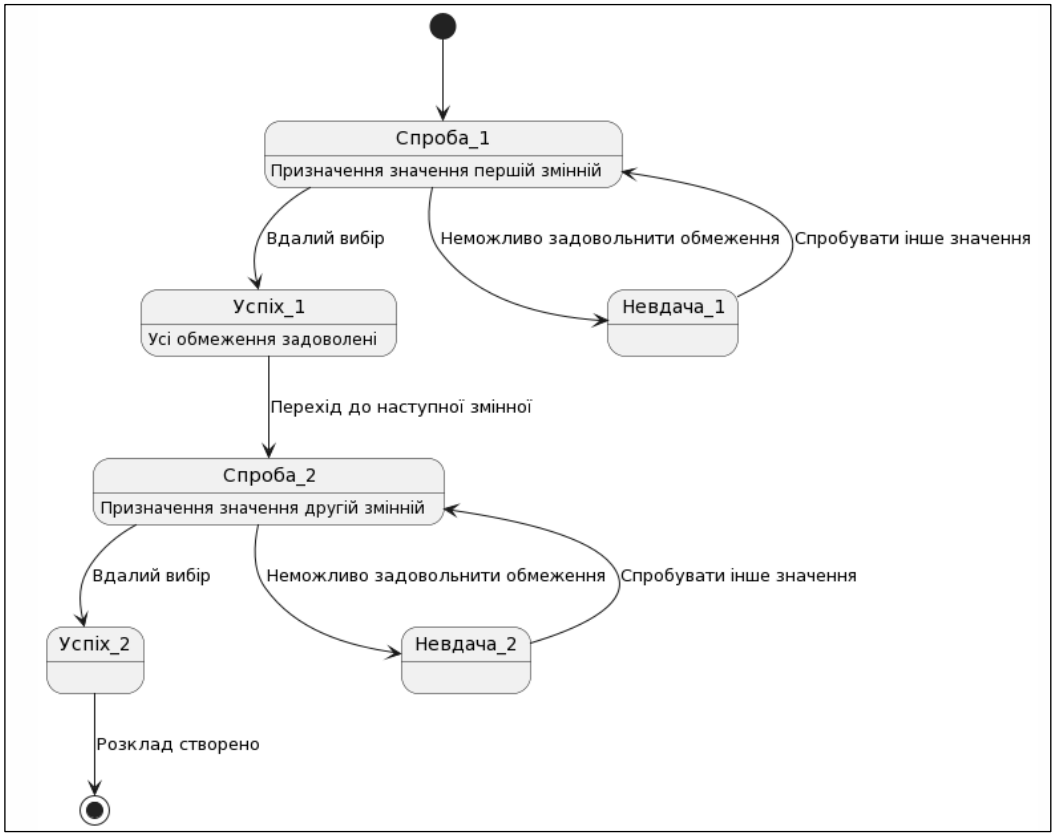

Рисунок 3.2 – Схема прикладного сценарію використання алгоритму бактрекінгу

3.3 Поєднання різних підходів у новий алгоритм

Ми зупинились на виборі алгоритму бактрекінгу та на генетичному алгоритмі з його можливостями мутації. Нижче пояснимо причини подібного вибору.

Ці алгоритми були обрані адже кожен день у нас оновлюється сет розкладу на наступний тиждень, розклад формується 7 днів на наступні 7, за допомогою ШІ формується приблизне навантаження на аналізі минулих тижнів, саме цей аналіз формує обмеження на персонал(які спеціалісти потрібні, спеціалісти якого досвіду потрібні, скільки працівників потрібно).

Сет кожного дня заповнюється побажаннями працівників що до бажаних годин роботи, із цього списку формується сет, та також на цей день формується сет від ШІ - скільки потрібно (які спеціалісти потрібні, спеціалісти якого досвіду потрібні, скільки працівників потрібно) потім алгоритмом (Генетичні алгоритми) зкрещуючи ці два сети щоб прийти до оптимального розкладу, який задовольнить і працівникам і рекомендованому сету із ШІ.

І такий алгоритм дій працює до тих пір поки розклад не почне конфліктувати – години працівників пересікаються або на якісь години працівників мало – тоді коли працівник заповнюватиме побажання – алгоритм CSP а саме алгоритм бектрекінгу буде проходитись по гілкам варіантів та якщо будуть грубі конфлікти він буде або не давати забронювати бажаний час працівнику, або пропонувати свій(на не заповнені години). Таким чином кожного дня розклад буде оновлюватись та саморегулюватись для фінальної побудови най оптимальнішого сценарію який знайде середне значення між побажаннями працівників та потребами закладу.

Фактично цей сценарій буде мати наступні етапи:

- аналіз і прогнозування ші: ші використовується для аналізу минулих даних та прогнозування майбутнього навантаження. це дозволяє сформувати базові обмеження для розкладу, включаючи потрібну кількість працівників, їхні спеціалізації та досвід;
- введення побажань працівників: працівники вносять свої побажання щодо графіку роботи, які стають частиною даних для формування розкладу;
- генетичні алгоритми для оптимізації: генетичні алгоритми можуть використовуватися для пошуку оптимального розкладу, який балансує між побажаннями працівників та потребами закладу;
- csp та бектрекінг для вирішення конфліктів: у випадку виникнення конфліктів або перетину часових слотів, алгоритм бектрекінгу в рамках csp може використовуватися для виявлення альтернативних варіантів розкладу, які не порушують обмежень;
- динамічне оновлення та саморегулювання: розклад постійно оновлюється і адаптується з урахуванням нових даних та змін, забезпечуючи гнучкість та відповідність актуальним умовам.

Використання цього підходу дозволить створювати розклади, які ефективно реагують на змінні потреби та очікування, мінімізуючи незадоволеність персоналу і підвищуючи загальну ефективність роботи закладу.

3.4 Розбір динамічного оновлення

Однією з ключових переваг нашого методу є здатність до динамічного оновлення. Ця особливість відіграє критичну роль у контексті "Intelligent Scheduling Systems", оскільки дозволяє системі швидко адаптуватися до змінних умов і потреб. Динамічне оновлення гарантує, що розклад залишається актуальним і ефективним навіть у випадку несподіваних змін, таких як відсутність працівників, зміни в попиті або інші оперативні виклики.

Ця здатність до динамічного оновлення базується на неперервному моніторингу даних та використанні алгоритмів машинного навчання та штучного інтелекту для аналізу та прогнозування оптимальних рішень у реальному часі. Такий підхід забезпечує високу гнучкість розкладу та здатність ефективно реагувати на будь-які зміни, забезпечуючи оптимальне розподілення ресурсів та задоволення потреб усіх сторін.

Для проектування функціоналу з динамічним оновленням у системі "Intelligent Scheduling Systems", розробники вирішили інтегрувати набір технологій і методик. В основі цього підходу лежить ідея створення адаптивної системи, яка може оперативно реагувати на зміни та оптимізувати розклади в реальному часі.

Перш за все, команда використовує базу даних, що постійно оновлюється, для збору інформації про робочі години, доступність персоналу, і вимоги до робочих змін. Ця база даних слугує основою для всіх подальших аналітичних процедур.

Далі, з використанням алгоритмів машинного навчання, система аналізує отримані дані для виявлення тенденцій та змін у патернах робочих навантажень. Ці алгоритми дозволяють прогнозувати майбутні потреби в персоналі та оптимальні конфігурації розкладу.

На основі аналізу та прогнозів, система використовує генетичні алгоритми або методи CSP для генерації оптимальних розкладів. Ці методи дозволяють ефективно комбінувати та адаптувати розклади, враховуючи нову інформацію та зміни у вимогах.

Ключовим аспектом динамічного оновлення є розробка модулю моніторингу, який постійно відстежує зміни у доступності персоналу, зміну попиту та інші критичні параметри. У разі виявлення змін, що вимагають корекції розкладу, система автоматично ініціює процес оптимізації.

Для забезпечення високого рівня адаптивності та ефективності, команда також інтегрує засоби штучного інтелекту, що дозволяють системі "вчитися" з досвіду та вдосконалювати свою здатність до прогнозування та оптимізації.

У підсумку, такий підхід до проектування системи з динамічним оновленням забезпечує гнучкість та ефективність розкладів, адаптуючись до поточних потреб персоналу та вимог закладу, і забезпечуючи високий рівень задоволення працівників і оптимальне використання ресурсів.

4 ЕКСПЕРЕМЕНТАЛЬНИЙ МЕТОД

4.1 Проектування методу

У цьому проекті ми об'єднали два основних підходи для розробки розкладу: Жадібний алгоритм та Генетичний алгоритм. Кожен з цих алгоритмів має свої сильні сторони та слабкості, і їх комбінація дозволяє досягти більш ефективного та збалансованого результату.

Жадібний алгоритм є одним з найпростіших та швидких методів для вирішення задач розподілу ресурсів. Його основний принцип полягає в тому, щоб завжди вибирати найкращий доступний варіант на кожному кроці. У контексті нашої задачі створення розкладу, жадібний алгоритм працює наступним чином:

Ініціалізація: Ми створюємо список працівників та їхні преференції щодо годин роботи кожного дня тижня. Крім того, визначаємо необхідну кількість працівників на кожну годину кожного дня.

Пріоритет годин: Ми визначаємо порядок годин для кожного дня, за яким буде відбуватися призначення працівників.

Призначення працівників: Для кожної години дня, виходячи з преференцій працівників, ми призначаємо працівників на роботу, поки не буде досягнуто необхідної кількості працівників.

Жадібний алгоритм ефективно вирішує проблему, коли існує чіткий набір пріоритетів, і його використання дозволяє швидко створити початковий розклад.

Генетичний алгоритм є більш складним методом, що імітує процеси природного відбору та генетичних мутацій.

Його основні кроки включають:

– ініціалізація популяції: генерується початковий набір розкладів, які слугуватимуть початковими кандидатами для оптимізації;
– оцінка приспособленості: кожен розклад оцінюється за допомогою функції, що вимірює, наскільки добре розклад задовольняє встановлені критерії;

- селекція: вибираються найкращі розклади для створення наступного покоління;
- схрещування: пари розкладів комбінуються для створення нових розкладів;
- мутація: вносяться невеликі випадкові зміни в розклади для підтримки різноманітності;
- повторення процесу: цей процес повторюється до досягнення оптимального розкладу або досягнення заданої кількості ітерацій;
- генетичний алгоритм дозволяє знаходити глобально оптимальні рішення, хоча він і вимагає значно більше обчислювальних ресурсів;
- для об'єднання цих двох підходів ми вирішили використовувати жадібний алгоритм для генерації початкового розкладу, який потім оптимізується за допомогою генетичного алгоритму. цей підхід дозволяє швидко отримати початковий розклад, який потім можна покращити до глобально оптимального за допомогою більш складного методу;
- початкова генерація розкладу: використовується жадібний алгоритм для швидкого створення початкового розкладу;
- оптимізація розкладу: початковий розклад передається генетичному алгоритму для подальшої оптимізації. генетичний алгоритм проводить оцінку, селекцію, схрещування та мутацію розкладів, поки не буде досягнуто оптимального результату.

Цей підхід дозволяє скористатися перевагами обох методів: швидкістю жадібного алгоритму та ефективністю генетичного алгоритму у знаходженні глобальних оптимумів.

Ми створили клас GreedyScheduleManager, який відповідає за генерацію початкового розкладу за допомогою жадібного алгоритму. Потім, у цьому ж класі, реалізували методи для оптимізації розкладу, які імітують основні кроки генетичного алгоритму.

Ціль: підготувати початкові дані про працівників та їхні преференції, а також необхідні кількості працівників на кожну годину.

Створення списку працівників: ми створюємо список об'єктів Employee, кожен з яких містить ім'я працівника та його преференції щодо годин роботи.

Визначення необхідної кількості працівників на кожну годину: Ми створюємо словник, де ключами є години роботи, а значеннями – необхідна кількість працівників.

Ціль: швидко створити початковий розклад, який задовольняє базові вимоги до кількості працівників на кожну годину.

Пріоритезація годин: визначаємо порядок годин, за яким буде відбуватися призначення працівників (наприклад, з понеділка з 9 до 17 години і так далі для інших днів).

Призначення працівників: для кожної години ми вибираємо працівників з найвищими преференціями на цю годину до досягнення необхідної кількості працівників.

Сортування працівників: працівники сортуються за їх преференціями для кожної години.

Призначення: працівники призначаються на роботу на відповідні години, поки не буде досягнуто необхідної кількості.

Оптимізація Розкладу (Генетичний Алгоритм), ціль: оптимізувати початковий розклад, щоб максимізувати задоволеність працівників та ефективність розкладу.

Оцінка поточного розкладу: використовуємо функцію приспособленості для оцінки, наскільки добре поточний розклад відповідає преференціям працівників і задовольняє потреби закладу.

Функція приспособленості враховує кількість годин, які працівник працює, та наскільки ці години відповідають його преференціям.

Селекція: вибираємо найбільш пристосовані розклади для створення нового покоління.

Турнірна селекція: випадково вибираємо групу розкладів і вибираємо з них найкращий.

Схрещування (кросовер): створюємо нові розклади, комбінуючи частини існуючих розкладів.

Одноточковий кросовер: вибираємо точку перетину і обмінюємося частинами між двома розкладами.

Мутація: вносимо невеликі випадкові зміни в розклади для підтримки різноманітності.

Випадкова зміна годин: Змінюємо преференції або призначення на декілька годин випадковим чином.

Повторення процесу: Повторюємо кроки селекції, схрещування та мутації до досягнення заданої кількості ітерацій або поки розклад не стане стабільним і оптимальним.

Показати кінцевий розклад для кожного працівника, який був оптимізований за допомогою генетичного алгоритму. Друкуємо розклад для кожного працівника, показуючи, які години вони будуть працювати кожного дня.\

Ми виводимо:

– ім'я працівника: виводимо ім'я працівника;
– розклад: виводимо години роботи для кожного дня тижня.

На рисунку 4.1 наведено приклад роботи наших алгоритмів разом.

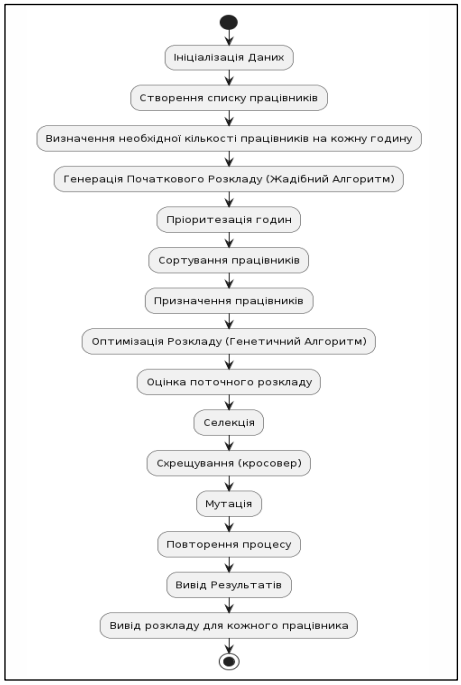

Рисунок 4.1 – Приклад поведінки алгоритмів

4.2 Архітектура системи

Далі на рисунку 4.2 наведемо приклад тестової архітектури нашого додатку та опишемо зв’язок елементів.

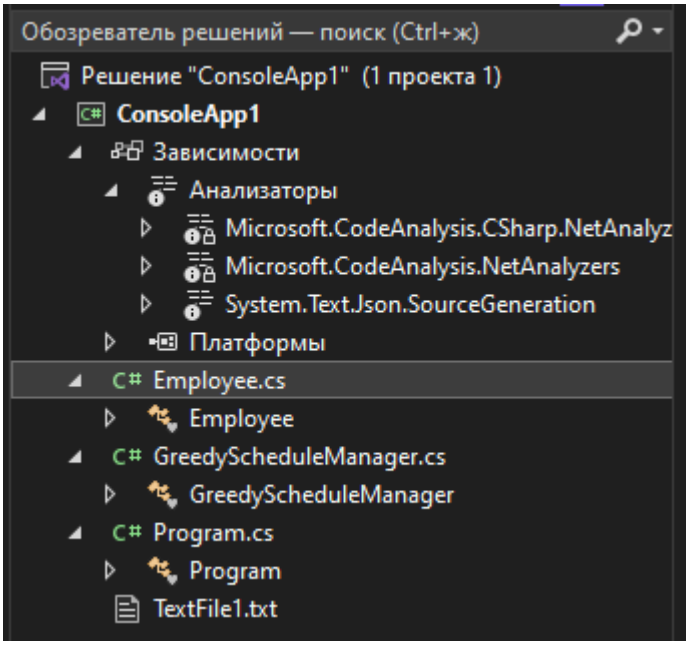

Рисунок 4.2 – Тестова архітектура додатку

Наше рішення має просту тестову файлова структуру:

- employee.cs:визначає клас employee, який слугує для зберігання інформації про окремих працівників, включаючи їх імена та робочі переваги;
- greedyschedulemanager.cs:містить клас greedyschedulemanager, який використовується для створення та оптимізації робочого розкладу за допомогою жадібного алгоритму. цей клас відповідає за управління розкладами працівників та оптимізацію цих розкладів;
- program.cs:головний вхідний файл програми, який використовує класи з інших файлів для запуску алгоритмів створення та оптимізації розкладу. він виконує ініціалізацію даних, викликає методи генерації та оптимізації розкладу, і в кінці виводить результати;
- textfile1.txt:текстовий файл, що міг би містити вхідні дані для програми, такі як інформація про працівників та їхні години переваг. у вашому

випадку, він може використовуватися для зчитування даних замість того, щоб визначати їх безпосередньо у коді.

Розглянемо зв'язок елементів:

– program.cs є основним драйвером програми, який координує діяльність між іншими компонентами;
– employee.cs визначає структуру даних для представлення працівників, яка використовується у program.cs та greedyschedulemanager.cs;
– greedyschedulemanager.cs використовує дані з employee.cs для створення та оптимізації розкладу;
– textfile1.txt може використовуватись для зчитування або зберігання даних про працівників, які потім обробляються іншими компонентами системи.

Ця структура дозволяє вашому додатку ефективно управляти даними про працівників, генерувати розклади на основі їхніх переваг, і оптимізувати ці розклади для досягнення оптимальних результатів (див.рис.4.3).

```
namespace ConsoleApp1
{
    Ссылок: 11
    class Employee
    {
        Ссылок: 3
        public string Name { get; set; }
        Ссылок: 10
        public Dictionary<string, int> Preferences { get; set; } = new Dictionary<string, int>();
        Ссылок: 9
        public Dictionary<string, int> AssignedHours { get; set; } = new Dictionary<string, int>();

        Ссылок: 5
        public Employee(string name)
        {
            Name = name;
        }
    }
```

Рисунок 4.3 – Модель користувача для обробки

Employee слугує для представлення працівників і зберігання відповідної інформації, яка потрібна для формування та оптимізації розкладу роботи.

Цей клас є ключовим у створенні гнучких та ефективних розкладів, які забезпечують врахування індивідуальних потреб працівників у вашому додатку.

Надалі реальними користувачами було заповнено тестовий файл із відповідними процентними бажаннями що до кожної години роботи(див. рис. 4.4).

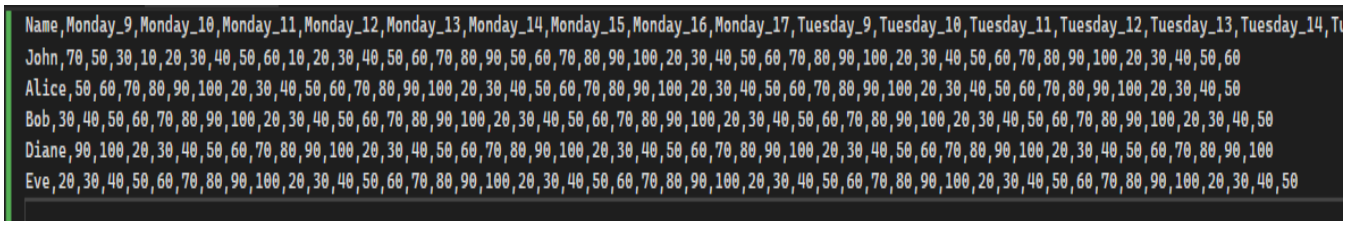

Рисунок 4.4 – Приклад тестових даних

Далі варто навести головний код алгоритму нашого застосунку та пояснити його роботу:

```
    foreach (var hour in hoursPriority)
            {
                int requiredEmployees =
requiredEmployeesPerHour.ContainsKey(hour) ? requiredEmployeesPerHour[hour]
: 0;
                var sortedEmployees = employees.OrderByDescending(e =>
e.Preferences.ContainsKey(hour) ? e.Preferences[hour] : 0).ToList();

                foreach (var employee in sortedEmployees)
                {
                    if (requiredEmployees == 0) break;

                    if (employee.Preferences.ContainsKey(hour) &&
employee.Preferences[hour] > 0)
                    {
                        employee.AssignedHours[hour] = 1;
                        requiredEmployees--;
                    }
                    else
                    {
                        employee.AssignedHours[hour] = 0;
                    }
                }
            }
            Console.WriteLine("Initial schedule generation complete.");
        }

        public void OptimizeSchedule()
        {
            Console.WriteLine("Starting optimization of schedule...");
            const int maxDailyHours = 40;

            foreach (var day in employees.First().AssignedHours.Keys)
            {
                int totalHours = employees.Sum(e => e.AssignedHours[day]);
```

```
                while (totalHours > maxDailyHours)
                {
                    foreach (var employee in employees)
                    {
                        if (employee.AssignedHours[day] > 0)
                        {
                            employee.AssignedHours[day]--;
                            totalHours--;
                            Console.WriteLine($"Reduced hours for
{employee.Name} on {day}.");
                            if (totalHours <= maxDailyHours) break;
                        }
                    }
                }
            }
            Console.WriteLine("Schedule optimization complete.");
        }

        public void PrintSchedules()
        {
            foreach (var employee in employees)
            {
                Console.WriteLine($"Employee: {employee.Name}");
                foreach (var day in employee.AssignedHours.Keys)
                {
                    Console.WriteLine($"{day}:
{employee.AssignedHours[day]} hours");
                }
                Console.WriteLine();
            }
        }
    }
```

Цей алгоритм починає роботу з ініціалізації даних, де визначається кількість потрібних працівників на кожну годину і відбувається сортування працівників за їхніми перевагами. В процесі генерації початкового розкладу, система працює з переліком годин за пріоритетністю, призначаючи працівників, чиї преференції вищі, і зменшуючи необхідну кількість працівників на кожну годину, поки потреби не будуть задоволені.

Далі відбувається оптимізація розкладу, де обчислюється загальна кількість робочих годин на день, і якщо ця кількість перевищує допустимий максимум, години роботи окремих працівників знижуються до тих пір, поки загальна кількість не опуститься до прийнятних меж.

Наприкінці процесу відбувається виведення остаточного розкладу для кожного працівника, показуючи, які години і на які дні їм призначено працювати,

що дозволяє забезпечити чітке відображення розподілу робочих годин згідно з індивідуальними перевагами і вимогами оптимізації.

4.3 Аналіз результатів

На рисунку 4.5 наведено результати відпрацювання алгоритму де видно що система встановила для кожного користувача персональні години.

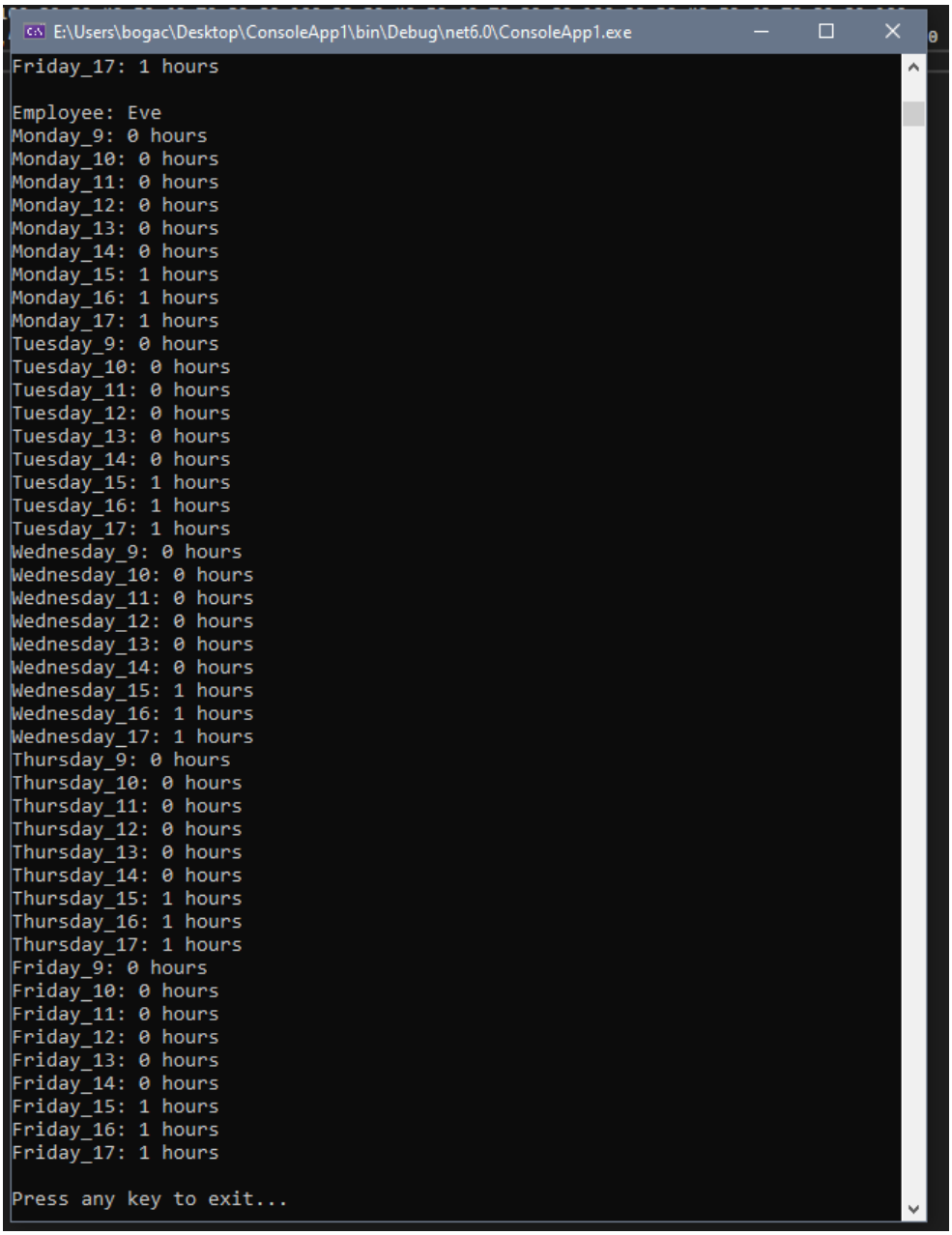

Рисунок 4.5 – Результати раскладу для кожного користувача

З наших результатів видно, що працівник на ім'я Eve має такий розклад роботи:

- більшість годин протягом тижня eve призначено працювати 1 годину, а деякі години вона не працює взагалі;
- це вказує на кілька можливих аспектів щодо алгоритму і вхідних даних;
- розподіл годин: алгоритм виконує призначення годин на основі преференцій eve, де вона має більшу кількість годин призначених у певні дні тижня. можливо, її преференції були вищими для тих годин, де їй призначено працювати, або ж вона могла бути єдиною доступною на ці години;
- низька кількість годин: існує багато годин, де eve не призначено жодної роботи, що може вказувати на недостатній попит на роботу або на високу конкуренцію серед працівників за ці години;
- оптимізація розкладу: залежно від того, як алгоритм враховує загальну кількість годин, які eve може працювати на день або тиждень, можливо відбувається обмеження годин, щоб не перевищити максимально допустимі обсяги.

4.4 Інтеграція зі машинним навчанням

Один зі способів використання нашого алгоритму генерації розкладу у машинному навчанні – застосування підкріплювального навчання для автоматичного вдосконалення призначення годин працівникам. Ось як це можна реалізувати:

Сценарій застосування підкріплювального навчання:

- середовище в вашому випадку – це всі можливі стани розкладу працівників, де кожен стан відповідає конкретному розкладу годин;
- визначення агента. агент – це алгоритм підкріплювального навчання, який приймає рішення про призначення годин працівникам. його мета – максимізувати загальне задоволення працівників і ефективність розкладу;
- визначення винагород. функція винагороди може бути визначена як комбінація загального задоволення працівників від розкладу та

дотримання робочих годин і потреб закладу. наприклад, винагорода може збільшуватися за кожного задоволеного працівника і зменшуватися за кожен випадок перевантаження або недостатнього залучення працівників;

- навчання агента. з використанням алгоритмів підкріплювального навчання, агент намагається знайти політику (серію дій), яка максимізує сумарну винагороду довгостроково. агент експериментує з різними призначеннями годин і навчається від кожної ітерації розкладу;
- впровадження та оцінювання. оптимізований розклад, сформований агентом, впроваджується в реальні умови. результати та відгуки працівників аналізуються для додаткового налаштування і покращення алгоритму.

Цей підхід дозволить вашій системі розкладу не тільки автоматично адаптуватися до змін у перевагах та доступності працівників, але й неперервно вдосконалюватися, використовуючи фідбек для покращення задоволеності персоналу і загальної продуктивності.

ВИСНОВКИ

У цій роботі ми провели глибокий аналіз різних методів для створення розкладів у контексті "Intelligent Scheduling Systems". Ми детально розглянули предметну область розкладів, необхідності їх оптимізації та їх створення.

Далі було проаналізовано ряд потенційно задовольняючих наші вимоги алгоритмів. Було розглянуто лінійне програмування, алгоритми жадібної оптимізації, генетичні алгоритми, алгоритми CSP та використання штучних нейронних мереж і машинного навчання. Кожен із цих підходів має свої переваги та обмеження в контексті розробки ефективних розкладів.

За результатами аналізу, ми прийшли до висновку, що оптимальним рішенням може бути комбінування алгоритму бектрекінгу з генетичними алгоритмами. Таке поєднання дозволяє використовувати переваги обох підходів: гнучкість бектрекінгу в розв'язанні комплексних задач з великою кількістю обмежень та потужність генетичних алгоритмів у пошуку оптимальних рішень у великих просторах пошуку.

Ця комбінація може підвищити шанси на знаходження більш ефективного та адаптивного розкладу, який враховує різноманітні потреби та обмеження у реальному часі. Таким чином у певному проміжку, розклад буде оновлюватись та саморегулюватись для фінальної побудови най оптимальнішого сценарію який знайде середне значення між побажаннями працівників та встановленими потребами закладу чи компанії.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. FastCompileExpression бібліотека \ Fast Compiler for C# Expression Trees and the lightweight LightExpression alternative. 2024, URL: https://github.com/dadhi/FastExpressionCompiler (дата звернення: 02.04.2024).

2. Optimization Algorithms in Project Scheduling - This resource from IntechOpen provides an in-depth look at various optimization algorithms used in project scheduling, including cost-related and quality-related objectives, as well as multi-objective project scheduling models. URL: https://www.intechopen.com/chapters/74156 (дата звернення: 04.04.2024).

3. Exploring the Potential of Genetic Algorithms for Optimizing Academic Schedules - A study from SpringerLink discusses the use of genetic algorithms to optimize academic schedules, with a focus on the School of Mechatronic Engineering. It includes practical results and methodologies for implementing these algorithms. URL: https://link.springer.com/article/10.1007/s00500-021-05916-3 (дата звернення: 17.04.2024). (дата звернення: 04.04.2024).

4. Using Genetic Algorithms in Optimizing Construction Material Delivery Schedules - This article from Emerald Insight examines how genetic algorithms can optimize delivery schedules in construction projects, emphasizing the reduction of costs and the efficiency of scheduling processes. URL: https://www.emerald.com/insight/content/doi/10.1108/14714170810846503/full/html (дата звернення: 10.04.2024).

5. A Hybrid Genetic Algorithm for Nurse Scheduling Problem - Published in the Journal of Healthcare Engineering, this paper explores a hybrid genetic algorithm addressing the nurse scheduling problem, taking into account factors like fatigue. URL: https://www.hindawi.com/journals/jhe/2021/6683765/ (дата звернення: 11.04.2024).

6. Improved Genetic Algorithm to Solve the Scheduling Problem of College English Courses - This study from the journal Complexity proposes enhancements to genetic algorithms for solving college course scheduling problems, demonstrating significant improvements in efficiency. URL:

https://www.hindawi.com/journals/complexity/2021/5512346/ (дата звернення: 12.04.2024).

7. Free Public Data Sets For Analysis, URL: https://www.tableau.com/learn/articles/free-public-data-sets (дата звернення: 15.04.2024).

8. FastCompileExpression бібліотека \ Fast Compiler for C# Expression Trees and the lightweight LightExpression alternative.? 2024р., URL: https://github.com/dadhi/FastExpressionCompiler (дата звернення: 17.04.2024).

9. Optimization Factors in Modeling and Testing Hardware and Semiconductor Defects by Dynamic Discrete Event Simulation \ Arabian, J. H., Видавництво: ХНУРЕ, 2009р., URL: https://openarchive.nure.ua/entities/publication/be918fba-8755-4d34-be6d-fc1464089d77 (дата звернення: 18.04.2024).

10. Falatiuk H., Shirokopetleva M., Dudar Z. Investigation of Architecture and Technology Stack for e-Archive System. 2019 IEEE International Scientific-Practical Conference Problems of Infocommunications, Science and Technology (PIC S&T), Kyiv,Ukraine,8-11October2019.2019.,URL: https://doi.org/10.1109/picst47496.2019.9061407 (дата звернення: 29.04.2024).

11. 14. DATA EXCHANGE MODEL IN THE INTERNET OF THINGS CONCEPT / I. Afanasieva et al. Telecommunications and Radio Engineering. 2019. Vol. 78,no. 10.P. 869–878.
URL: https://doi.org/10.1615/telecomradeng.v78.i10.30 (date of access: 05.05.2024).