НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
Інститут прикладного системного аналізу

Кафедра системного проектування

До захисту допущено:

Завідувач кафедри

_____ А.І. Петренко

«___»_____2021 р.

Дипломна робота

на здобуття ступеня бакалавра
за освітньо-професійною програмою «Інтелектуальні сервіс-орієнтовані розподілені обчислювання»
спеціальності 122 «Комп'ютерні науки»
на тему: «Автоматизація складання розкладу занять в навчальних закладах з використанням алгоритму системи мурашиних колоній»

Виконала:
студентка IV курсу, групи ДА-71
Вовк Ірина Сергіївна _____

Керівник:
Доцент, к.т.н.
Безносик Олександр Юрійович _____

Консультант з економіки:
Доцент, к.е.н.
Рощина Надія Василівна _____

Рецензент:
Професор каф. АПЕПС ТЕФ, д.т.н., професор
Аушева Наталія Миколаївна _____

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Вовк Ірина Сергіївана_____

Київ – 2021

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Інститут прикладного системного аналізу
Кафедра системного проектування

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 122 «Комп'ютерні науки»

Освітньо-професійна програма «Інтелектуальні сервіс-орієнтовані розподілені обчислювання»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ А.І. Петренко
«___»_____2021 р.

ЗАВДАННЯ

на дипломну роботу студенту
Вовк Ірина Сергіївна

1. Тема роботи «Автоматизація складання розкладу занять в навчальних закладах з використанням алгоритму системи мурашиних колоній», керівник роботи Безносик Олександр Юрійович, доцент, к.т.н., затверджені наказом по університету № 1148-с від 25.05.2021 р.

2.Термін подання студентом роботи 17.06.2021

3. Вихідні дані до роботи

Методологія складання розкладу для учбових закладів, мова програмування C++, графічний фреймворк Qt

4. Зміст роботи

1. Формалізація задачі складання розкладу. Складання математичної моделі задачі.
2. Порівняння існуючих програмних засобів для вирішення проблеми автоматизованого складання розкладу.

3. Порівняння існуючих алгоритмів для вирішення NP-повних задач.
4. Опис алгоритму для автоматизованого складання розкладу.
5. Результати роботи програмного продукту. Тест точності та ефективності запропонованого алгоритму.

5. Перелік ілюстративного матеріалу (із зазначенням плакатів, презентацій тощо)

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Економічний	Рощина Н. В		

7. Дата видачі завдання 03.02.2021

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка
1	Отримання завдання	03.02.2021	
2	Збір інформації	29.03.2021	
3	Ознайомлення з літературою і підготовка теоретичної частини роботи	11.04.2021	
4	Аналіз вимог завдання, вибір методів і засобів розв’язання поставленої задачі	26.04.2021	
5	Розробка алгоритма для автоматизованого складання розкладу	04.05.2021	
6	Розробка програмної реалізації	14.05.2021	
7	Тестування розробленої системи	22.05.2021	
8	Розробка економічної частини дипломного проекту	25.05.2021	
9	Оформлення дипломної роботи	04.06.2021	
10	Отримання допуску до захисту та подача роботи в ДЕК	09.06.2021	

Студент Ірина ВОВК

Керівник Олександр БЕЗНОСИК

АНОТАЦІЯ

Дипломна робота: 171 с., 51 рис., 6 табл., 1 дод., 10 джерел.

АВТОМАТИЗОВАНЕ СКЛАДАННЯ РОЗКЛАДУ, АЛГОРИТМ СИСТЕМ МУРАШИНИХ КОЛОНІЙ, NP - ПОВНІ ЗАДАЧІ.

Предмет дослідження — методи вирішення NP-повних задач.

Об’єкт дослідження — автоматизоване складання розкладу в навчальних закладах.

Мета роботи — автоматизація етапу формування розкладу шляхом розробки ефективного алгоритму складання розкладу в навчальних закладах.

Методи дослідження — використання евристичних алгоритмів для автоматизованого складання розкладу.

Створено алгоритм для автоматизованого складання розкладу в навчальних закладах.

Проведено огляд основних підходів для вирішення NP-повнх задач.

Створено програмний продукт з використанням побудованого алгоритму.

Оцінено точність та швидкість роботи створеного прогармного продукту.

ABSTRACT

Thesis: 171 pages, 51 figures, 6 tables, 1 appendices, 10 sources.

AUTOMATED SCHEDULE, ANT COLUMN SYSTEMS ALGORITHM , NP - COMPLETE.

Subject of the work — methods for solving NP-complete problems.

Object of work — automated scheduling in educational institutions.

Aim of the work —automation of the stage of scheduling by developing an effective algorithm for scheduling in educational institutions.

Research methods — use of heuristic algorithms for automated scheduling.

Research methods — use of heuristic algorithms for automated scheduling.

An algorithm for automated scheduling in educational institutions has been created.

An overview of the main approaches to solving NP-complete problems.

A software product was created using the proposed algorithm.

The accuracy and speed of the created software product are estimated.

ЗМІСТ

ВСТУП

Етап календарного планування є важливою частиною як у створенні нового проекту, так і в складанні плану робіт вже довго функціонуючої установи чи виробництва. Адже саме від цього етапу залежить наскільки ефективно будуть використані людські та матеріальні ресурси і головним чином - час.

Особливе місце серед задач календарного планування займає задача складання розкладу в навчальних установах. Саме від його оптимальності залежить якість подання матеріалу викладачами, засвоєння отриманих знань учнями та студентами та розподіл ресурсів навчального закладу. Це все є вагомими чинниками, що здійснюють безпосередній вплив на якість підготовки майбутніх фахівців.

Етап календарного планування - трудомістка задача, яка потребує врахування одночасно великої кількості критеріїв та їх взаємного впливу. Її автоматизація могла би значною мірою спростити адміністрування та управління проектом чи установою та оптимізувати його роботу.

Теорія розкладів почала вивчатись ще в шестидесятих роках минулого століття і незважаючи на велику кількість алгоритмів та програмних продуктів, створених для вирішення задач планування, не втратила свою актуальність і сьогодні. Задачі теорії розкладів відносять до класу NP-повних, тому не існує ефективного алгоритму, для знаходження найбільш оптимального рішення. Точні алгоритми гарантовано знаходять найоптимальніше рішення, проте при великому розмірі вибірки, що є типовим для прикладних задач, час, витрачений на пошук такого рішення, може досягати порядку доби або навіть більше. Такі значні витрати часового ресурсу не є прийятними. Тому для задач з високою обчислювальна складністю знайшли застосунок наближені алгоритми, які за прийнятний час можуть побудувати рушення, що є наближеним до найбльш оптимального, проте з деякою втратою точності. Постала задача збереження балансу між часом роботи алгоритма та відхиленням знайденого рішення від найоптимальнішого.

Метою даної бакалаврської роботи є автоматизація етапу формування розкладу шляхом розробки ефективного алгоритму складання розкладу в навчальних закладах.

Основні задачі алгоритма:

1. врахування правил та обмежень, що накладаються на розклад навчальною установою
2. дотримання навчальної методології
3. впорядкування навчального процесу найбільш сприятливим чином, як для надання знань, так і для їх засвоєння

Цілі бакалаврського дослідження:

1. огляд існуючих методів вирішення задачі календарного планування
2. побудова математичної моделі для визначення якості розкладу занять в навчальних закладах
3. розробка ефективного алгоритму складання розкладу для навчальних закладів
4. створення програмного продукту для автоматизації процесу складання розкладу

1 ЗАДАЧА СКЛАДАННЯ РОЗКЛАДУ ДЛЯ НАВЧАЛЬНИХ ЗАКЛАДІВ

1.1 Постановка задачі

Задано кінцеву кількість часових інтервалів (таймслотов), яка дорівнює сумарній кількості навчальних годин всіх груп. Необхідно розмістити навчальні предмети (лекції і практичні заняття) таким чином, щоб задовільнити найбільшій кількості накладених обмежень.

1.2 Математична модель задачі

Нехай у навчальному закладі існує n навчальних груп. Введемо наступні позначення:

P — множина викладачів

L — множина лекцій

E — множина практик

A — множина аудиторій

T — множина таймслотів

Q — множина критеріїв, яким повинен відповідати розклад

$S = L \cup E$ — множина всіх предметів

За кожним предметом $s_i \in S, 1 \leq i \leq |S|$ закріплюється один таймслот $t_j \in T, 1 \leq j \leq |T|$, з чого слідує, що множина рішень є бієкцією $S \to T$. За кожним таймслотом $t_j \in T, 1 \leq j \leq |T|$ закріплюється одна аудиторія $a_k \in A, 1 \leq k \leq |A|$.

Розглянемо критерії, яким має задовольняти розклад. Їх можна умовно розділити на 2 групи: обов'язкові і бажані.

До обов'язкових можна віднести:

1. Виконання навчального плану
2. Кількість студентів на парі не повинно перевищувати місткість аудиторії
3. Заняття, що мають особливі вимоги (наприклад комп'ютерний клас), повинні проходити в спеціально обладнаних для цього аудиторіях
4. В один момент часу один викладач може проводити одне заняття

5. В один момент часу в одній аудиторії може проходити одне заняття в однієї групи (якщо це не загальна лекція потоку)
6. Максимальна кількість пар в день не повинна перевищувати вказану кількість С.
7. В один день у однієї групи по одному предмету повинно бути не більше 1 практичного і не більше 1 лекційного заняття

До бажаних критеріїв можна віднести:

1. Побажання викладачів про час проведення занять
2. Мінімізація вікон у викладачів
3. Мінімізація вікон у студентів
4. Лекційні заняття повинні передувати практичним або лабораторним
5. Бажано проводити лекційні та практичні заняття по одному предмету в один день
6. Розміщення лабораторних або практичних робіт на першій або останній парі

Нехай $\delta_h = \begin{cases} 1, \text{якщо h-й критерій виконується} \\ 0, \text{в іншому випадку} \end{cases}$

Оскільки критерії не рівнозначні, введемо ваговий коефіцієнт c_t, різний для кожного предмета.

Введемо функцію $m(s,t)$ оцінки розміщення предмета $s \in S$ в таймслоті $t \in T$.

$$m(s_i, t_j) = \sum_{h=0}^{|Q|} c_h * \delta_h \tag{1.1}$$

Нехай існує деякий алгоритм g, що вирішує дану задачу. g - приймає на вхід множини G, A, S, P і формує впорядковану множину типу $\{t_i, s_i, a_i\}$ для кожної групи. Нехай R - множина рішень, що були знайдені алгоритмом g.

Якість розкладу(одного рішення, знайденого алгоритмом g) визначається сумою оцінок розміщення s_i в t_j. Таким чином, розв'язок задачі скаладння розкладу зводиться до наступної задачі дискретної оптимизації:

$$\sum_{i,j \in R} m(s_i, t_j) \rightarrow max \tag{1.2}$$

1.3 Огляд існуючих програмних продуктів для вирішення задачі складання розкладу в навчальних закладах

На даний момент існує багато програмних продуктів для вирішення завдання календарного планування в навчальних закладах. Проте жодне з готових рішень не відповідає всім вимогам, що накладаються. Недоліком може бути час роботи, недостатня ступінь автоматизації або взагалі її відсутність.

Розглянемо детальніше декілька готових комерційних програмних продуктів, їх переваги та недоліки.

1. Docendo

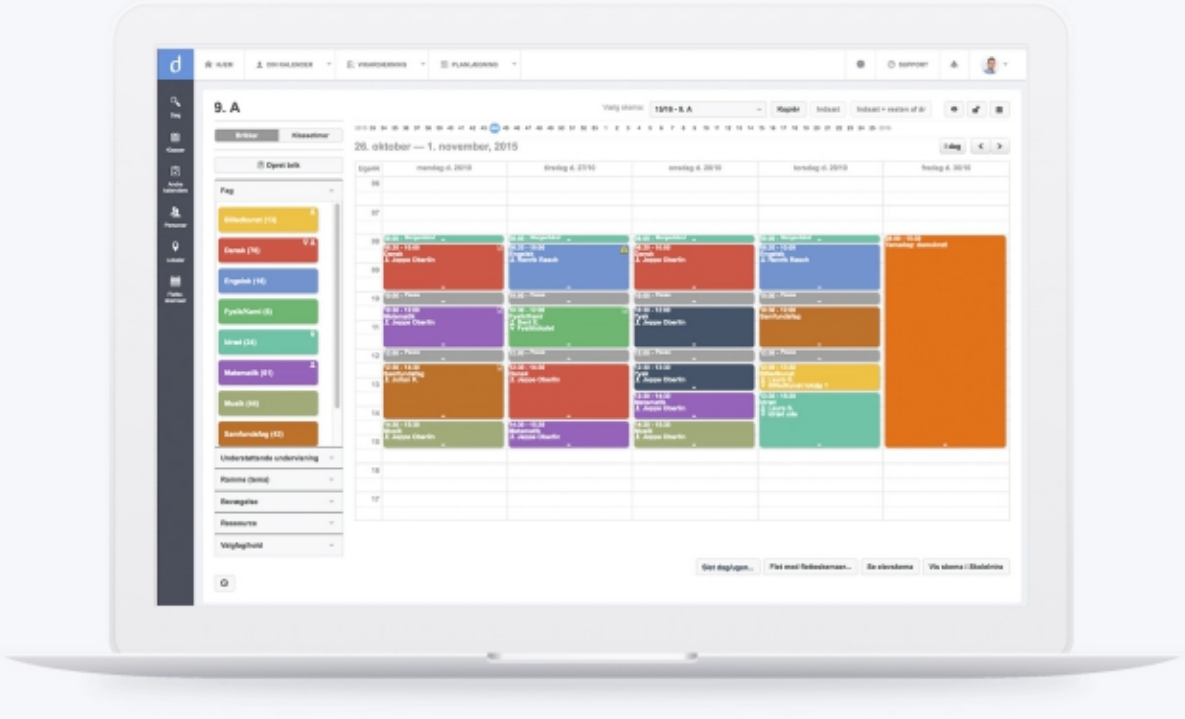

Рисунок 1.1 – Інтерфейс Docendo

Планер з функціоналом для організації навчального процесу в навчальних закладах. Надає можливість інтерактивного складання розкладу та поширення його в мережі.

Основним недоліком є відсутність алгоритму автоматизованого складання розкладу. Натомість пропонується вручну розміщувати навчальні години з підказками про накладені обмеження в часі.

2. QuickSchool

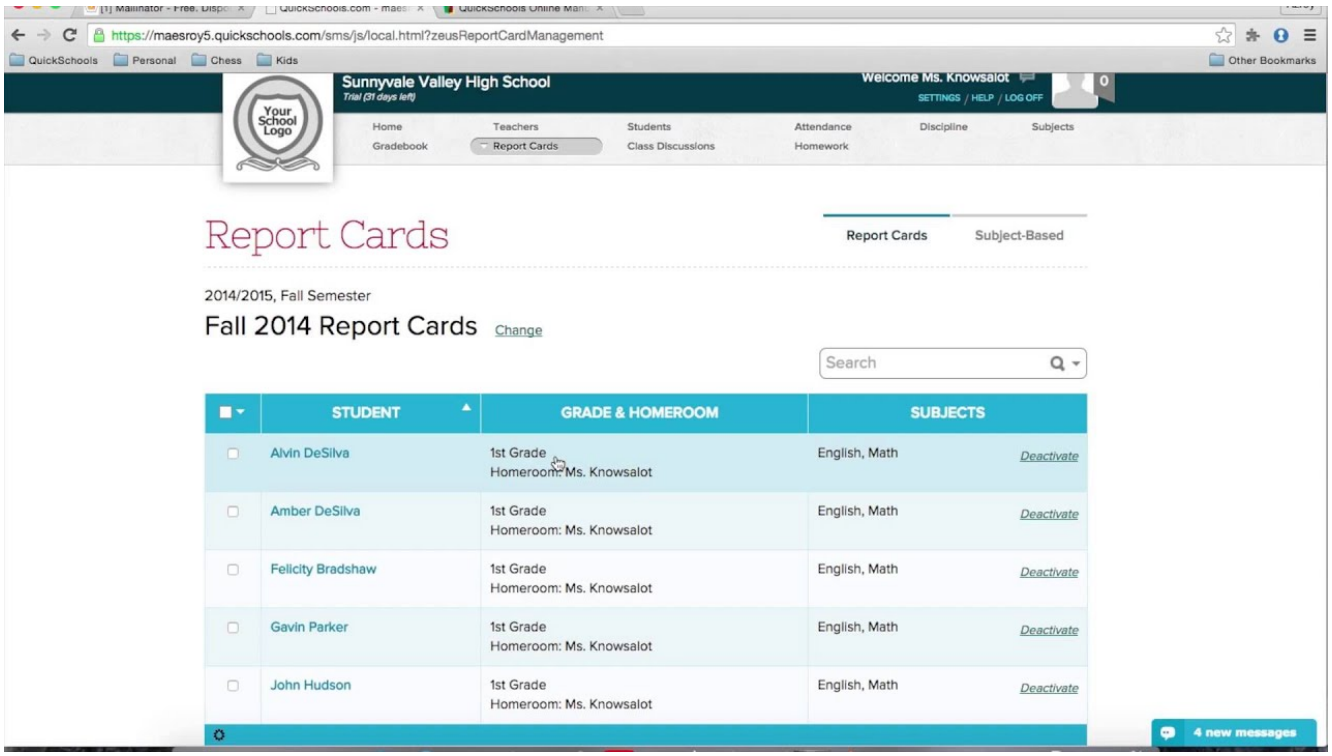

Рисунок 1.2 – Інтерфейс QuickSchool

Веб додаток, з більш розширеним функціоналом для складання розкладу, ніж попередній приклад. Має функцію автоматизованого складання розкладу, що задовільняє обов’язковим умовам. Наявна додаткова функуція ведення статистики успішності студентів.

Основним недоліком даного програмного забезпечення є відсутність автоматизованого пошуку найбільш оптимального рішення, з урахуванням бажаних критеріїв.

3. aSc Timetables

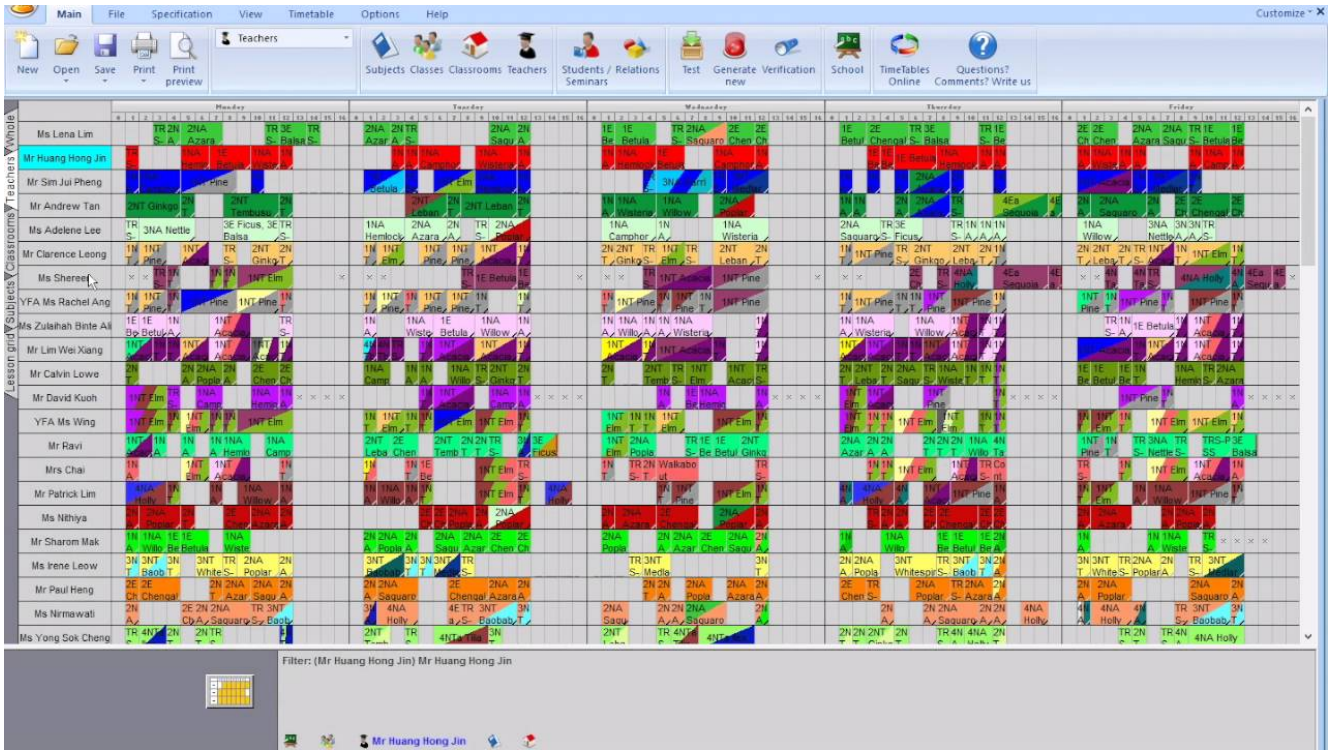

Рисунок 1.3 – Інтерфейс aSc Timetables

Програмне забезпечення для автоматизованого складання розкладу в навчальних закладах. Надає змогу вносити зміни в складений розклад, задовільняє обов’язковим та бажаним вимогам.

Основним недоліком aSc Timetables є швидкість роботи.

Проаналізувавши програмні продукти, що представлені на ринку для вирішення проблеми календарного планування, сформулюємо загальні вимоги до програмного продукту:

1. автоматизоване знаходження найбільш оптимального розв’язку задачі складання розкладу навчального закладу
2. можливість вносити зміни в готовий розклад, не порушуючи накладені обмеження та умови
3. можливість поширення готового розкладу в мережі або експорту в зручний для користувача формат
4. швидкість пошуку розв’язку задачі

5. зручність використання

1.4 Теоретичні відомості

В даному підрозділі наведено основні поняття теорії складності алгоритмів, що будуть застосовані.

1.4.1 Часова складність алгоритма

Алгоритм вирішує конкретну задачу протягом $O(T(n))$, якщо для заданого екзампляра задачі i довжиною $n = |i|$ з його використаннямЮ можна отримати розв'язок за час $O(T(n))$. $O(T(n))$ - називають часовою складністю алгоритма.

Задачу називають вирішуваною за поліноміальний час, якщо існує алгоритм, який дозволяє її вирішити за час $O(n^k)$ для деякої константи k.

1.4.2 Клас складності P

Визначимо клас складності P як множину конкретних задач, що вирішуються за поліноміальний час.

1.4.3 Алфавіт

Алфавіт $\sum$ - це кінцева множина символів.

1.4.4 Мова

Мова L, що визначена на множині $\sum$, є довільною множиною строк, що складаються з символів множини $\sum$. Мова всіх строк, заданих над множиною $\sum$, включаючи пусту строку ϵ позначається $\sum^*$.

1.4.5 Алгоритм верифікації

Алгоритм A з двома аргументам, один з яких є звичайна вхідна строка x, а інший бінарна строка y. Двухаргументний алгоритм A верифікує вхідну строку x, якщо існує такий сертифікат y, що $A(x, y) = 1$. Мова верифікована алгоритмом A, представляє собою множину

$$L = \{x \in \{0,1\}^* : \exists x \in \{0,1\}^* : A(x, y) = 1\}$$

1.4.6 Клас складності NP

Клас складності NP - це клас мов, які можна верифікувати за допомогою алгоритма з поліноміальним часом роботи. Точніше кажучи, мова L належить класу NP тоді і тільки тоді, коли існує алгоритм A з двома вхідними параметрами і поліноміальним часом роботи, а також константа c, така що

$$L = \{x \in \{0,1\}^* : \exists y, |y| = O(|x|^c), : A(x,y) = 1\}$$

При цьому кажуть, що алгоритм A верифікує мову L за поліноміальний час.

1.4.7 Приводимість задач класа NP за поліноміальний час

Мова L_1 приводиться за поліноміальний час до мови L_2 ($L_1 \leq_P L_2$), якщо існує функція, що вираховується за поліноміальний час $f : \{0,1\}^* \to \{0,1\}^*\}$, така що для всіх $x \in \{0,1\}^*$ $x \in L_1$ тоді і тільки тоді, коли $f_2(x) \in L_2$.

Функцію f називають функцією приведення, а алгоритм F з поліноміальним часом роботи, що вираховує функцію f - алгоритмом приведення.

1.4.8 NP - повнота

Мова $L \subseteq \{0,1\}^*$ є NP-повною якщо

1. $L \in NP$
2. $L` \leq_P L$ для кожного $L` \in NP$

Якщо мова L задовільняє властивості 2, проте не обов'язково задовільняє властивості 1, кажуть що L - NP- складний.

1.5 Виснови до розділу 1

Задача складання розкладу є одним з основних етапів календарного планування. Для її вирішення необхідно одночасно враховувати велику кількість критеріїв та їх загальний вплив одне на одного. Це робить етап складання розкладу одним з найбільш трудомістких. Автоматизація могла би значним чином спростити цей процес.

Хоча проблема автоматизації складання розкладу не нова, досі не існує програмного продукту, що цілком задовільнив би всім накладеним потребам та мав у своїй основі ефективний за часом та оптимальний за точністю алгоритм.

Отже, метою даної дипломної роботи є створення ефективого алгоритму для автоматизованого складання розкладу та розробка програмного продукту, що задовільняє більшості накладених потреб.

2 ОГЛЯД ІСНУЮЧИХ МЕТОДІВ ВИРІШЕННЯ ЗАДАЧІ. ОПИС ЗАПРОПОНОВАНОГО АЛГОРИТМУ ДЛЯ ВИРІШЕННЯ ЗАДАЧІ СКЛАДАННЯ РОЗКЛАДУ

2.1 Огляд можливих альтернативних рішень задачі автоматизованого складання розкладу

Загалом, алгоритми, що застосовуються для вирішення NP - повних задач поділяються на 2 класи:

1. точні алгоритми - гарантовано знаходять найбільш оптимальний розв’язок, проте їх складність, зі збільшенням об’єму даних, росте за експоненціальним законом. До точних алгоритмів відноситься повний перебір (brute-force) та наближені до нього. Такі алгоритми недоцільно використовувати в прикладних задачах.
2. наближені алгоритми - знаходять розв’язок задачі, наближений до найбільш оптимального, проте є ефективними за часом роботи.

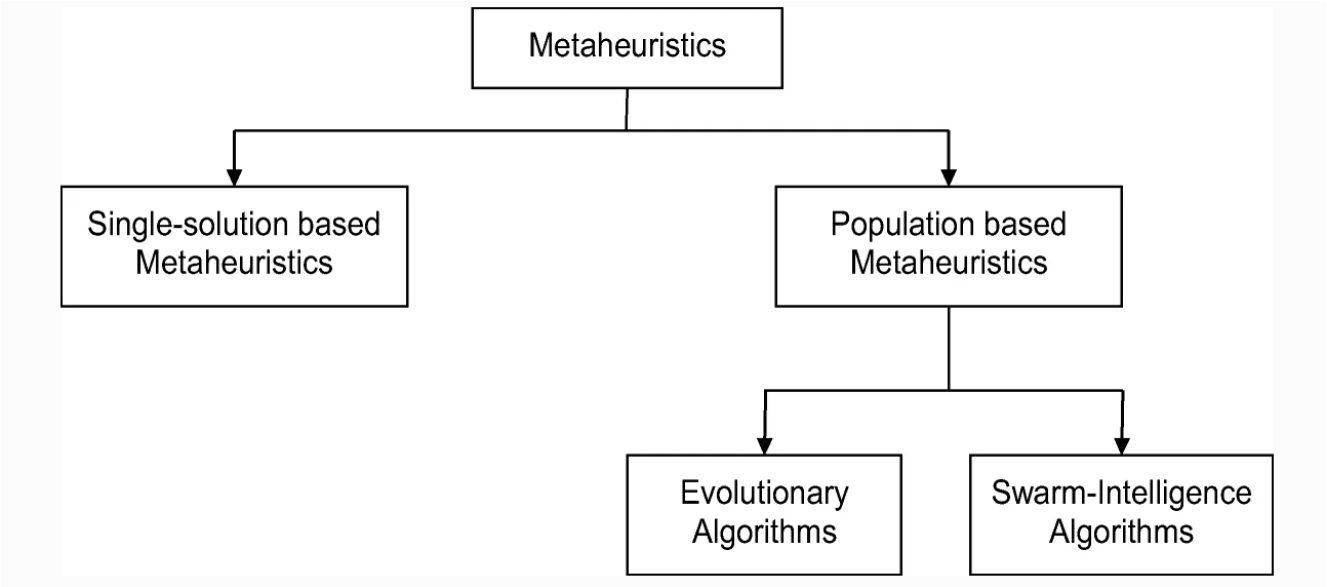

Рисунок 2.1 – Підмножини класу метаевристичних алгоритмів

Клас наближених алгоритмів має назву евристичного. Алгоритм, що поєднує декілька евтристик має назву — метаевристичний. Більшість метаевристичних алгоритмів походять від біологічних процесів, принципів

ройового інтелекту та законів фізики. У множині метаевристичних алгоритмів,опираючись на метод складання розв'язку, можна виділити дві підмножини: орієнтовані на єдине рішення (single based solution), орієнтовані на популяцію (population based) . Така класифікація наведена на рис. 2.1.

Метаевристичні алгоритми орієнтовані на єдине рішення - будують одне рішення - кандидат та за допомогою алгоритмів локального пошуку вдосконалюють його. Такий підхід допомагає зменшити ймовірніть "застрягання" рішення в локальному оптимумі.

На основі популяційної метаевристики в процесі побудови рішення генерується безліч рішень-кандидатів. З яких надалі, з використанням деякої функції оцінки, обираються найкращі.

Як було зазначено раніше, задачі теорії розкладів відносять до класу NP-повних, це означає що задачу складання розкладу можна звести до будь-якої задачі класу NP - повних.

Найбільше досліжень ефективності алгоритмів проводилось для класичної NP-повної задачі комівояжера (TSP). Проведемо огляд алгоритмів, що використовуються для вирішення задач цього класу відносно TSP.

Опираючись на дослідження *"Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem", Université Libre de Bruxelles, Marco Dorigo, Luca Maria Gambardella 1996 року.* Розглянемо порівняння наступних наближених алгоритмів:

1. Genetic algorithm (GA)
2. Evolutionary programming (EP)
3. Simulated annealing (SA)
4. Ant colony system (ACS)

Далі буде більш детально описано кожен з наведених вище алгоритмів.

2.2 Генетичний алгоритм

Генетичний алгоритм (GA) — це один з найвідоміших метаевристичних алгоритмів, який натхненний процесом біологічної еволюції. Генетичний

алгоритм імітує дарвінівську теорію виживаня найсильніших у природі. Він був запропонований Г.Х. Голланд в 1992 р.

Основними етапами GA є створення хромосом, вибір кращих кандидатів за допомогою фітнес функції та оператори, натхненні біологічним процесами. Голанд також представив новий елемент, а саме ”Інверсію”, який зазвичай використовується в реалізаціях GA. Як правило, хромосоми приймають формат двійкового рядка. У хромосомах кожен локус (конкретне положення на хромосомі) має два можливі стани - 0 і 1. Хромосоми розглядаються як точки в просторі рішення. Вони обробляються за допомогою генетичних операторів шляхом ітеративного заміщення його популяції. Фітнес - функція використовується для присвоєння значення всім хромосомам у популяції.

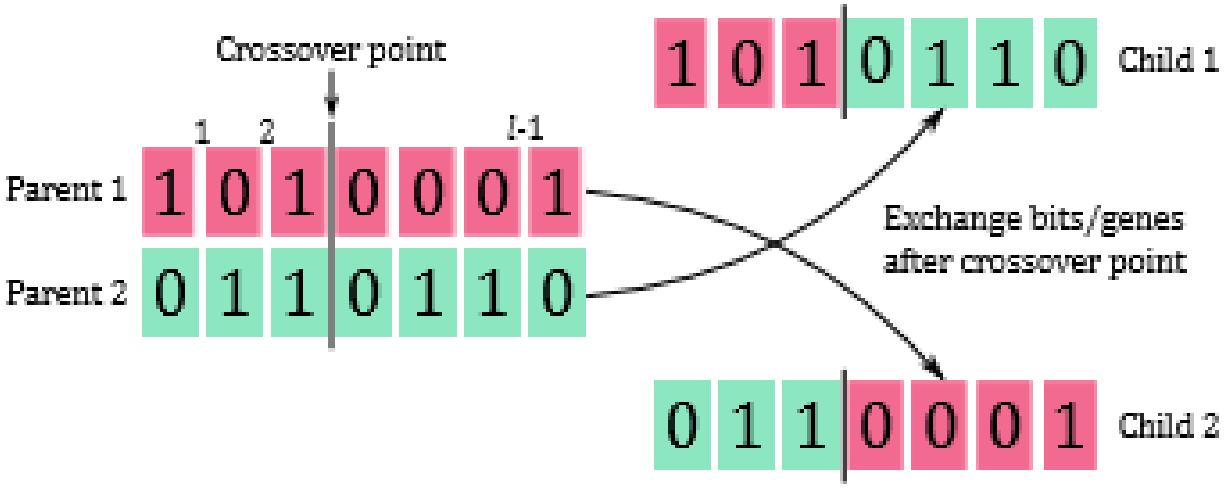

Рисунок 2.2 – Іллюстрація роботи генетичного алгоритма

Операторами GA є мутація та кросовер. Оператор кроссоверингу працює наступним чином: вибирається випадковий локус, який змінює підпослідовності між хромосомами, щоб створити нову. В такому випадку хромосоми на яких використовується цей оператор називають батьківськими, а рузультат його роботи — дочірньою хромосомою або хромосомою-спадкоємцем.

При мутації деякі біти хромосом будуть випадковим чином переміщені на основі ймовірнісного вибору.

До переваг генетичного алгоритму можна віднести:

1. Генетичний алгоритм можна комбінувати з іншиим точними алгоритмами для покращення знайденого кращого рішення-кандидата або декількох найкращих кандидатів перед наступною ітерацєю кросоверу.
2. Може бути використаний автоматичний підбір необхідних коефіцієнтів(напр. ймовірність мутації, параметри оператора кроссовера і т.д.) шляхом самобалансування, базуючись на динаміці покарщення рішення.
3. Є ефективними для вирішення задач, що важко формалізуються, адже не потребують попередньої експертної оцінки.

До недоліків генетичного алгоритму можна віднести:

1. Не існує ефективних параметрів для завершення роботи алгоритма.
2. Має тенденцію ”застрягати” в локальних оптимумах функції
3. Не є ефективним для мінімізації унімодільної функції, адже не може використовувати інформацію про градієнт функції.

2.3 Еволюційне програмування

Еволюційне програмування — одна з основних парадигм еволюційних алгоритмів. Воно схоже на генетичний алгоритм, за виключенням того, що алгоритм, який ми повторюємо, є фіксованим, але його параметри відкриті для оптимізації. Вперше такий підхід був застосований Лоуренсом Дж. Фогелем у США в 1960 році для того, щоб використовувати модельовану еволюцію як навчальний процес, спрямований на створення штучного інтелекту.

Еволюційне програмування просто копіює механізм природного відбору. Основні кроки,та їх послідрвнсіть зображено на рис. 2.3.

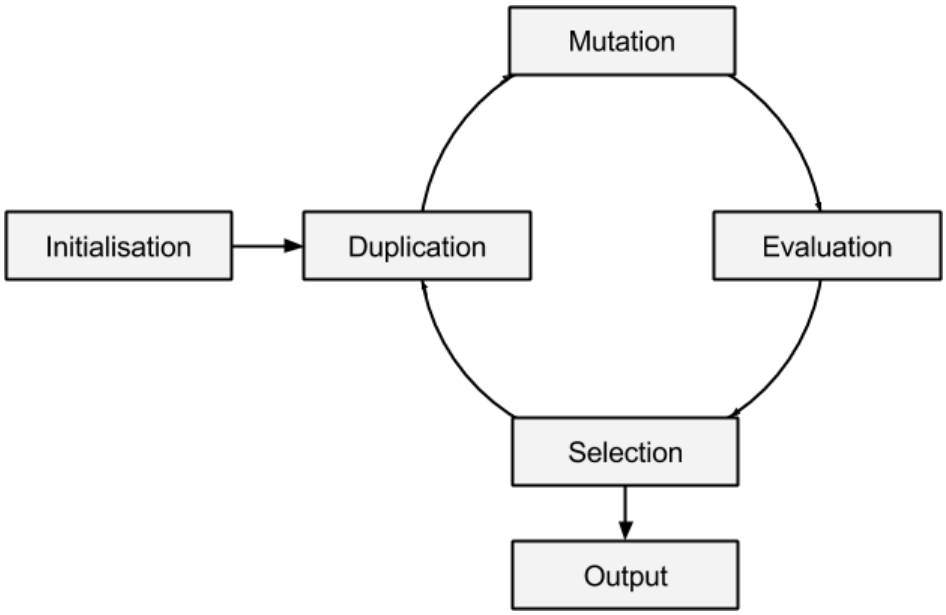

Рисунок 2.3 – Послідовність основних етапів еволюційного програмування

Розглянемо детальніше основні етапи еволюційного програмування:

1. Ініціалізація. Необхідно згенерувати почтакове рішення. Цей етап є важливим, адже різні початкові рішення можуть призвести до абсолютно різних результатів.
2. Дублювання. Створення великої кількості копій поточного рішення.
3. Мутація. Кожна копія мутує випадковим чином. Ступінь мутації є критично важливою, оскільки вона контролює швидкість, з якою протікає еволюційний процес.
4. Оцінка. Отримані екземпляри оцінюються за допомогою визначеної задалегіть функції оцінки.
5. Відбір. Після того, як екземпляри були оцінені, можна дублювати лише найкращих, щоб створити наступне покоління.
6. Вихідні дані. Еволюція - це ітераційний процес, у будь-який момент його можна зупинити, щоб отримати вдосконалену (або незмінну) версію попереднього покоління.

До переваг еволюційного програмування можна віднести:

1. Є ефективними для вирішення задач, що важко формалізуються, адже не потребують попередньої експертної оцінки.
2. Практично будь-який параметр алгоритма може бути змінений, відповідно до потреб задачі, що вирішується.

До недоліків еволюційного програмування можна віднести:

1. Не існує метода для визначення момента зупинки алгоритма.
2. Має тенденцію ”застрягати” в локальних оптимумах фінкції.
3. Результат роботи алгоритма значним чином залежить від етапу ініціалізації.

2.4 Алгоритм імітації відпалу

Алгоритм імітації відпалу був розроблений в 1983р. і є одним із перших метаевристичних алгоритмів. Цей алгоритм був натхненний фізичними явищами, що відбуваються при затвердінні речовин, таких як метали.

Як це відбувається в інших методах без використання похідних, алгоритм імітації відпалу запобігає ”застряганню” в локальних мінімумах за допомогою випадкової величини, в даному випадку, вираженої через ланцюг Маркова. Імітований відпал вносить зміни в рішення для поліпшення цільової функції, але також зберігає рішення, які, незважаючи на неефективність, відповідають певним критеріям. У задачі мінімізації використовуються будь-які кращі модифікації, що зменшують цільову функцію J, проте рішення-кандидати, що збільшують J, також включаються до набору рішень з імовірністю p, що називається ймовірністю переходу:

$$p = exp[-\Delta E/k_B T]$$

де k_B — постійна Больцмана,

T - параметр управління, температурою процесу відпалу,

ΔE - рівень енергії, який пов’язаний із зміною цільової функції через константу γ :

$$\Delta E = \gamma \cdot \Delta J$$

Приймається чи не змінюється рішення, вирішується випадковим порогом r, так що $p \geq r$. В процесі оптимізації T зменшується, і система охолоджується, як визначено коефіцієнтом охолодження,

$$T(t) = T_0 \cdot \alpha^t$$

де $t = 1, 2, ..., t_f$ і $\alpha \in [0, 1]$

Як наслідок вибір вихідного T_0 сильно впливає на рішення. Дуже великі значення спричинили б прийняття всіх рішень, тоді як дуже низькі значення не сприяли б різноманітності, як це відбувається в методах сходження на пагорби. Отже, велика T передбачає високоенергетичні системи та труднощі з досягненням мінімумів, а низька T означає низькоенергетичні системи, які можуть потрапити в локальні мінімуми. Правильний вибір T повинен мати можливість спочатку шукати глобальні мінімуми, а пізніше, коли система охолоджується, сходитися до рішення з належною точністю. Критерій зупинки включений у t_f, який обмежує кількість ітерацій або з точки зору вдосконалення цільової функції.

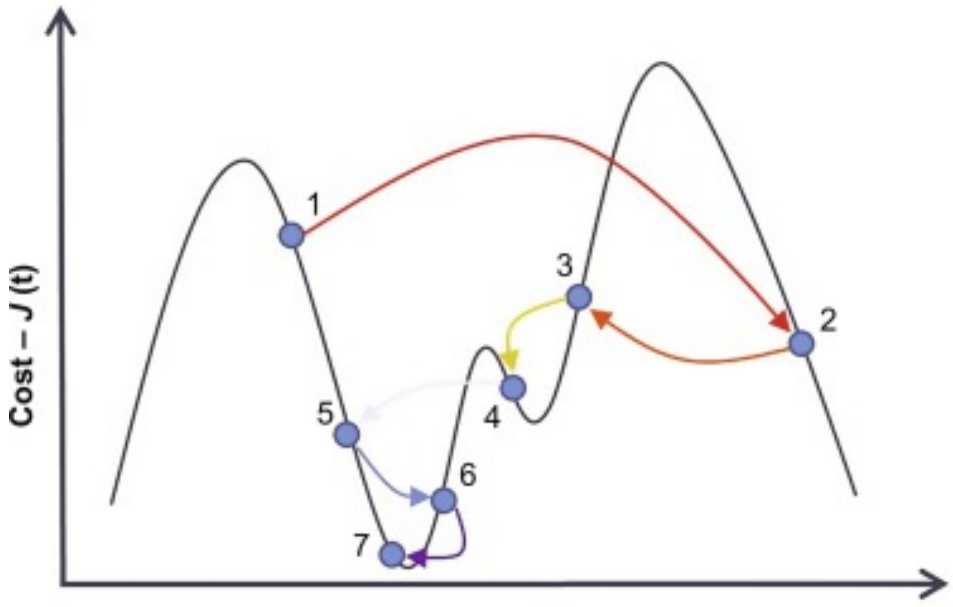

Рисунок 2.4 – Іллюстрація роботи алгоритма моделювання відпалу на 7 ітерацій.

Рисунок 2.4 іллюструє приклад модельованої збіжності відпалу в семи ітераціях. Спочатку енергія системи велика через великі значення T, здійснюючи

пошук по всьому простору в ітераціях 1–3. Оскільки система «охолоджується», пошук рішення зменшує варіативність кандидатів, як у 3–5 і, нарешті, низькі температури дозволяють точно сходитися до оптимуму (рішення), після ітерації 7.

Перевагами модельованого відпалу є його легка реалізація та можливість знайти глобальний оптимум навіть після знаходження локального мінімуму, оскільки він приймає рішення, які гірші за найкращого кандидата. Крім того, він може забезпечити задовільні результати при відносно малій кількості ітерацій, що робить його придатним для контролю в режимі реального часу.

2.5 Алгоритм систем мурашиних колоній

Алгоритм систем мурашиних колоній - евристичний ймовірнісний алгоритм для пошуку найоптимальнішого шляху в графі. Даний алгоритм має біологічне походження і є одним з методів ройового інтелекту.

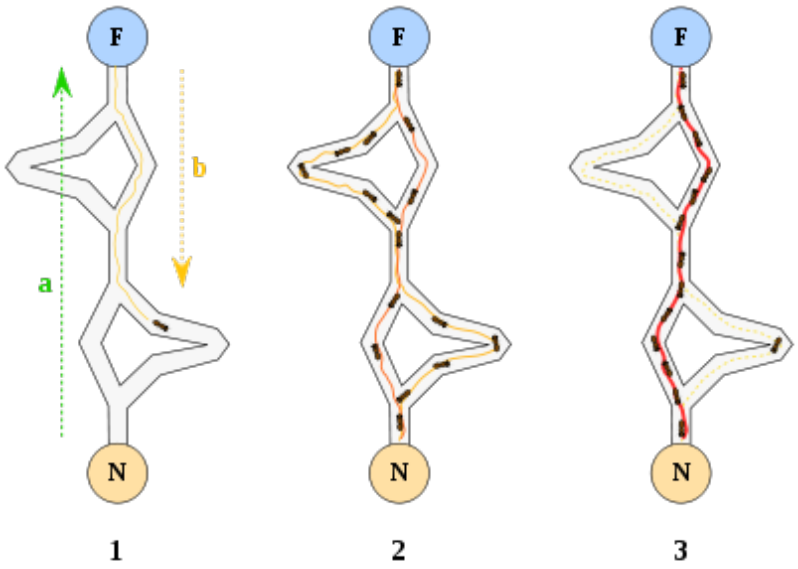

Рисунок 2.5 – Основна ідея роботи алгоритма мурашиних колоній

Алгоритм грунтується на поведінці мурашиної колонії в реальному житті. Мурахи бігають по стежках в пошуках найкоротшого шляху до джерела їжі. На стежці, по якій проходить мураха, залишається речовина (феромон), яка утворює так званий феромонний слід. Ця речовина допомагає орієнтуватися іншим мурахам в моменті вибору стежки. Чим більше мурах пройшли по стежці

— тим більше феромона на ній залишилося — тим вище ймовірність, що наступна мураха теж вибере цю стежку. Таким чином, через деякий час мурахи сформують найбільш оптимальний шлях до джерела їжі. Цей принцип зображено на рис. 2.5.

Формалізуємо описану вище ідею. Введемо деяку кількість агентів (мурах), які використовують загальну пам’ять у вигляді феромонного сліду. На кожній ітерації агенти переміщаються по, побудованому для задачі, графу незалежно один від одного. Так, в кінці кожної ітерації ми отримаємо набір рішень, що дорівнює кількості мурах.

Алгоритм ACS має 3 основні правила:

1. Правило вибору наступної вершини
2. Правило локального оновлення феромона
3. Правило глобального оновлення феромона

Для можливості оцінки того, яку вершину найбільш оптимально обрати наступною, ребра графа мають бути зваженими. Вага кожного ребра визначаються кількістю феромона, залишеного на ньому агентами, та евристичною інформацією, яка визначається виконанням накладених на розклад обмежень.

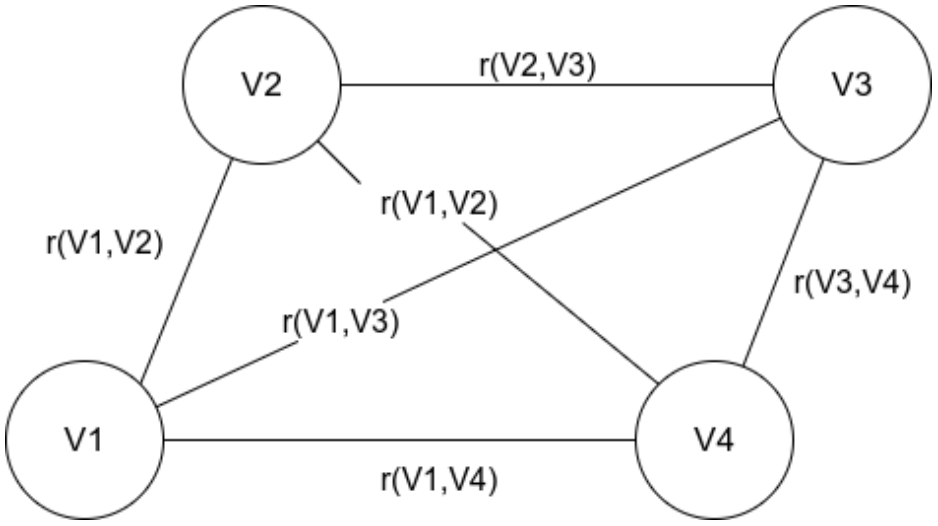

Рисунок 2.6 – Граф G_1

Нехай G_1 — зважений неорієнтованный повнозв'язний граф з вершинами $\{V_1, \ldots, V_n\}$. $r(V_i, V_j)$ — вага ребра, яке з'єднує вершину i і вершину j. При цьому $r(V_i, V_j) = r(V_j, V_i)$.

Вибір наступної вершини V_j для k-го агента, який знаходиться в вершині V_i відбувається за наступним правилом:

$$c_k = \begin{cases} argmax_{u \in J(V_i)} \left\{ \left[\tau(r,u)^\alpha \cdot [\eta(r,u)^\beta] \right] \right\}, q \leq q_0 \\ p_k(V_i, V_j) \end{cases} \tag{2.1}$$

де $q \sim U(0,1)$,

$q_0 \in [0,1]$ — параметр, заданий на этапі ініціалізації алгоритма

$J(V_i)$ - множина не відвіданих k -м агентом вершин

Алгоритм ACS - є ймовірнісним. Ймовірність вибора k-м агентом ребра $\{V_i, V_j\}$:

$$p_k(V_i, V_j) = \begin{cases} \frac{[\tau(V_i,V_j)]^\alpha \cdot [\eta(V_i,V_j)]^\beta}{\sum_{u \in J_k(V_i)}^{|J_k|} [\tau(V_i,u)]^\alpha \cdot [\eta(V_i,u)]^\beta}, V_j \in J_k(V_i) \\ 0, V_j \notin J_k(V_i) \end{cases} \tag{2.2}$$

Правило перехода між вершинами, котре випливає з (2.1) і (2.2) називається правилом випадкової порпорціональності. Такий принцип вибору наступної вершини допомагає зберегти баланс між використанням найкращого варіанту переходу($q \leq q_0$), і дослідженням нових вершин ($q > q_0$).

Під час пошуку рішення, агент проходить по ребрах графа, а значить феромонний слід, залишений на ребрі, змінюється за таким правилом:

$$\tau(V_i, V_j) \leftarrow (1 - \rho) \cdot (\tau(V_i, V_j) + \rho \cdot \Delta\tau(V_i, V_j) \tag{2.3}$$

де $\rho \in [0,1]$ — коефіцієнт випаровування феромона

$\Delta\tau(s_i, s_j) = \tau_0$ — параметр, заданий на етапі ініціалізації.

В кінці ітерації феромонний слід оновлюється за формулою:

$$\tau(s_i, s_j) \leftarrow (1 - \rho) \cdot \tau(V_i, V_j) + \alpha \cdot \Delta\tau(V_i, V_j), \tag{2.4}$$

де

$$\Delta\tau(V_i, V_j) = \begin{cases} \frac{1}{L_{it_b}}, V_i, V_j \in B \\ 0 \end{cases} \tag{2.5}$$

де L_{itb} — найкраще рішення, знайдене на ітерації.

B— множина ребер, які входять в найкраще рішення, знайдене на ітерації.

α— коефіцієнт випаровування феромнона.

Таким чином, в першому випадку, значення феромона для ребер, які входять в краще рішення на даній ітерації, збільшується на $\Delta\tau(s_i, s_j)$,для інших ребер — зменшується.

2.6 Обґрунтування вибору базисного, для дипломної роботи, алгоритма

Тестування ефективності проводилось за такими критеріями: найкращий розв'язок знайдений алгоритмом (у порівнянні з накращим можливим розв'язком), кількість спроб для пошуку найкращого розв'язку.

Табл. 1 – Результати тесту ефективності алгоритмів для проблеми TSP

Problem name	ACS	GA	EP	SA	Optimum
Eli50 (50-city problem)	425 (427.96) [1,830]	428 (N/A) [25,000]	426 (427.86) [100,000]	443 (N/A) [68,512]	425 (N/A)
Eli75 (75-city problem)	535 (542.37) [3,480]	545 (N/A) [80,000]	542 (549.18) [325,000]	580 (N/A) [173,250]	535 (N/A)
KroA100 (100-city problem)	21,282 (21,285.44) [4,820]	21,761 (N/A) [103,000]	N/A (N/A) [N/A]	N/A (N/A) [N/A]	21,282 (N/A)

Для проведнння тесту було використано 3 вибірках для проблеми TSP:

1. Eli50 - 50 міст
2. Eli75 - 75 міст
3. KroA100 - 100 міст

Результати тестування наведено в таблиці вище.

З огляду на результати тестування, можемо зробити висновок, що найбільш ефективно використовувати алгоритм систем мурашиних колоній (ACS).

2.7 Опис побудованого для задачі складання розкладу алгоритма

В розділі 1 було описано основні позначення для можин, які будуть використовуватися. Наведемо більш делальний опис атрибутів елементів кожної множини.

1. P - множина викладачів

   Кожен елемент даної множини має такі атрибути: Ф.І.О, кількість робочих годин за тиждень, матриця побажань викладача про час проведення заняття.

2. S - множина занять(лекцій, практичних, лабораторних)

   Кожен елемент даної множини має такі атрибути: назва, тип(лекція/практика), тип класу, необхідного для проведення заняття (з додатковим обладненням/без додаткового обладнання), частота проведення заняття (щотижневий/ один раз на 2 тижні).

3. A - множина аудиторій

   Кожен елемент даної множини має такі атрибути: номер кімнати, номер корпуса(будівлі), місткість аудиторії, наявність спеціального обладнання (напр. для лабораторних робіт).

4. G - множина груп

   Кожен елемент даної множини має такі атрибути: індекс групи, курс, кількість студентів, граф.

Опираючись на описані вище атрибути множин, можемо отримати більш чітку специфікацію задачі автоматизованого формування розкладу. Ця

специфікація необхідна для побудови алгоритма найбільш ефективним чином. Тож, враховуючи обґрунтування вибору алгоритма систем мурашиних колоній, що було наведено в попередньому підрозділі, можемо чітко та покроково сформувати алгоритм, що буде застосовано для даної задачі.

2.8 Використання алгоритма систем мурашиних колоній для задачі складання розкладу в навчальних закладах

Використанная ACS для задачі складання розкладу можна описати, викристовуючи наступну діаграму:

Ця діаграма показує ієрархію відношень між компонентами ACS. Розглянемо більш детально роль та функціонал кожного компонента.

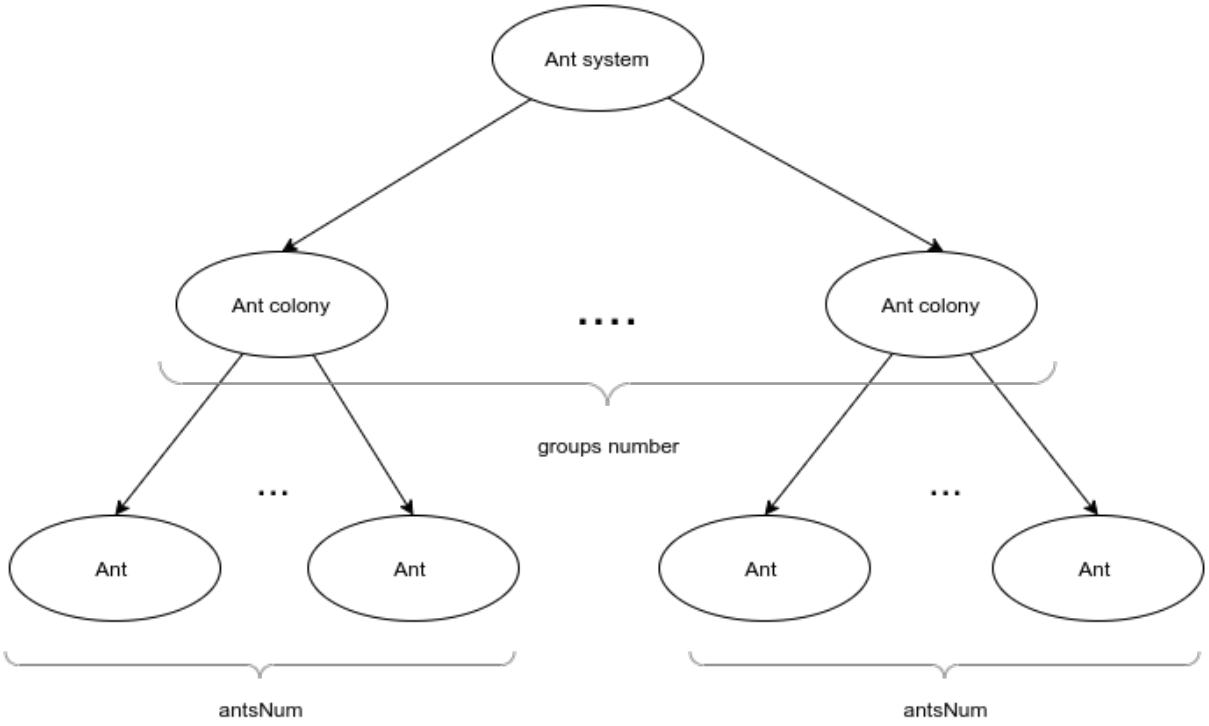

Рисунок 2.7 – Ієрархія відношень між компонентами ACS

1. Ant system — створення колоній, формування та оцінка рішень-кандидатів
2. Ant colony — створення агентів(мурах), оновлення феромонного сліда (локальне і глобальне)
3. Ant — формування розкладу для однієї групи

Для кожної групи створюється окрема колонія. Кількість агентів в кожній колонії визначається параметром $antsNum$, та визначає кількість рішень-кандидатів, що формуються на кожній ітерації. Таким чином, якщо об'єднати розклад, складений і-тим агентом в кожній колонії - отримаємо одне повне рішення-кандидат.

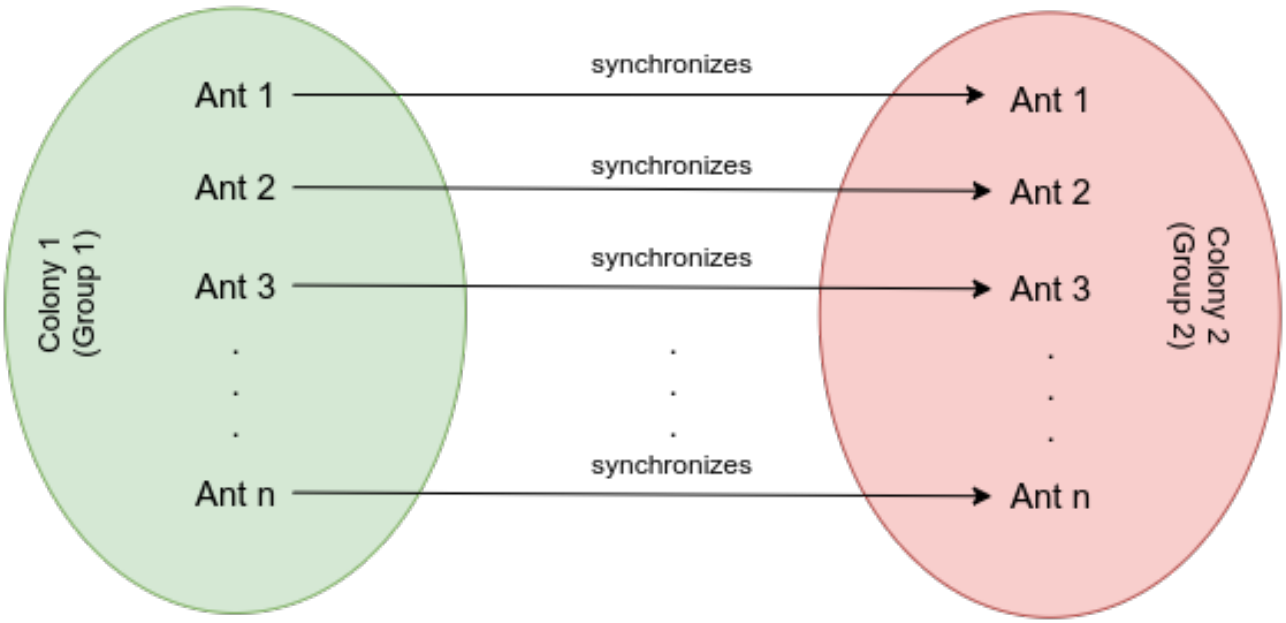

Рисунок 2.8 – Синхронізація між агентами різних колоній

Оскільки задача складання розкладу є задачею впорядкування, доцільним буде представити її у вигляді повнозв'язного, неорієнтованого графа. Вершинами якого є пари типу $\{s_i, p_i\}, s_i \in S, p_i \in P$. Кількість вершин графа дорівнює кількості академічних робочих годин групи за один тиждень. Також, вводиться додаткова вершина V_0 для старта роботи алгоритма. Ця вершина не враховується в результуючому рішенні і потрібна для того, щоб агенти мали вибір вершии для першої пари в розкладі.Та вершина GAP - для врахування "вікон" в розкладі. Вершина GAP,навідміну від інших вершин, в результуючому розкладі може враховуватись більше 1 раза. Кількість таких врахувань для $g_i \in G$ групи дорівнює

$$h_{total} - h_{work_i}$$

де h_{total} - кількість робочих годин в тижні

h_{work_i} - кількість робочих годин для g_i групи.

Для можливості розпаралелення алгоритма, граф складається окремо для кожної групи. Приклад побудованого для однієї групи графу зображено на рис. 2.9.

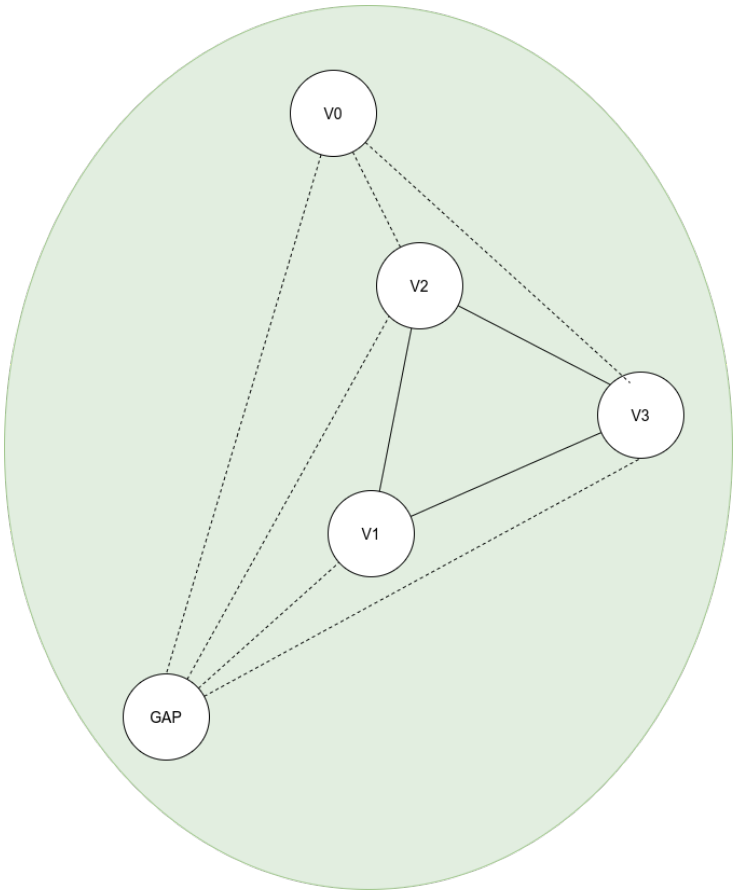

Рисунок 2.9 – Приклад побудови графа задачі складання розкладу для однієї навчальної групи

Проте, необхідно врахувати те, що деякі заняття можуть проводитись загально для декількох групу (напр. загальні лекції потока). Відповідні вершини в графі з'єднуються загальним ребром. Таким чином, отримаємо m-мірний граф, мірність якого дорівнює кількості груп навчального закладу, для якого складається розклад. Приклад такого графа наведено на рисунку 2.10.

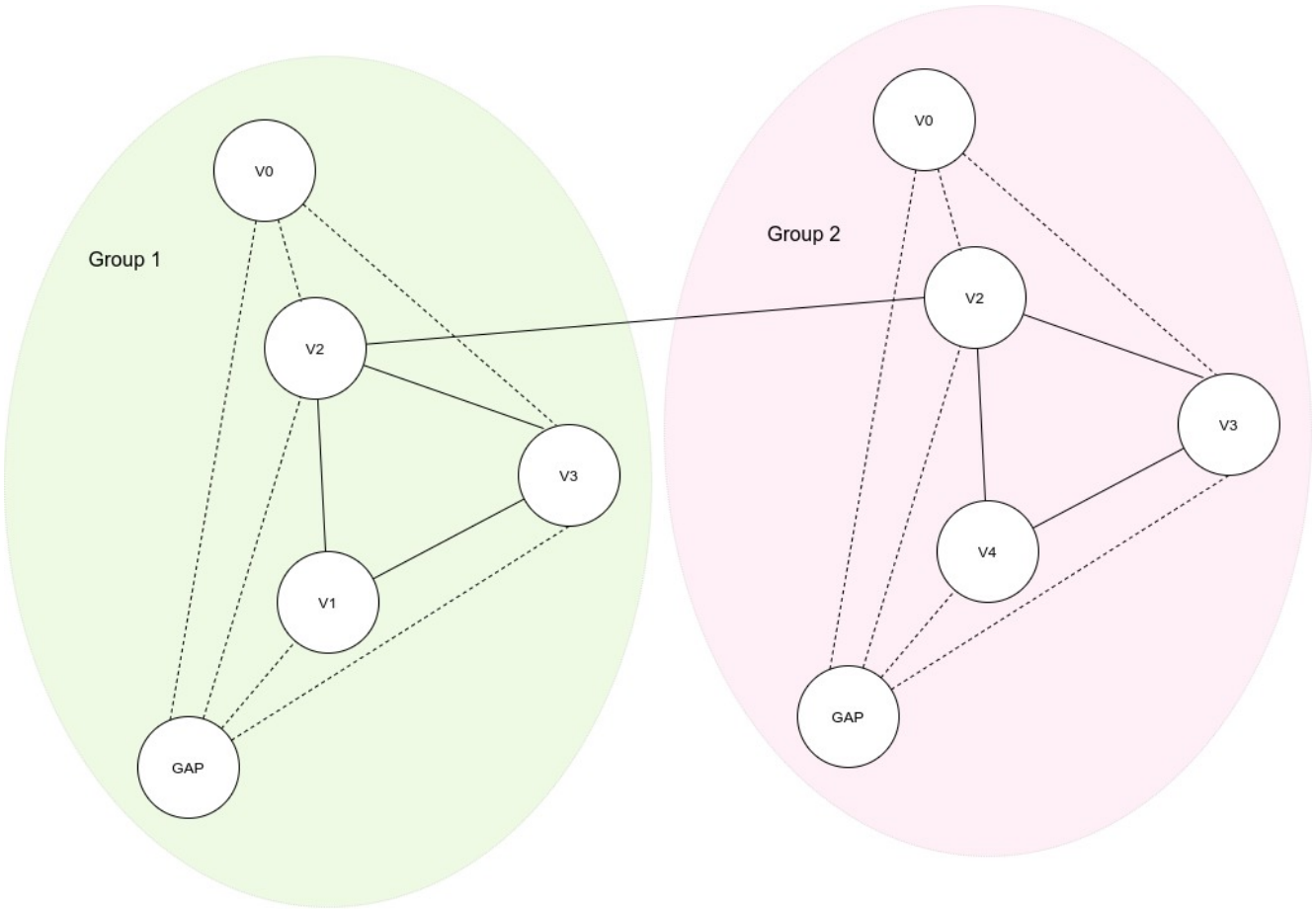

Рисунок 2.10 – Приклад побудови графа задачі складання розкладу

На рисунку зображений граф задачі складання розкладу для 2 груп, у кожної з яких є 2 незалежні вершини V_1, V_3 та одна загальна V_2.

Особливу увагу необхідно приділити розгляду питання синхронізації та задоволенню обов'язковим обмеженням. Кожен і-тий агент в кожній з колоній синхронізований в питаннях вибору наступної вершини та аудиторії.

Це виключає

- можливість проведення одиним викладачем одночасно декількох занять
- присутність однієї групи одночасно на декількох заняттях
- присутність декількох груп в одній аудиторії одночасно

Вибір аудиторії для проведення заняття відбувається по принципу першої - підходящої. Тобто, всі аудиторії діляться на групи за такими критеріями

– місткість аудиторії (лекторії/ кабінети)
– наявність спеціального обладанання (кабінети/лабораторії)

За типом вершини та кількістю студентів в групі визначається належність шуканої аудиторії до першого типу, а за потребою спеціального обладнання для проведення заняття — до другої.

Таким чином, можемо сформулювати основні кроки алгоритма:

1. Ініціалізація параметрів алгоритма
2. Введення користувачем даних про предмети, викладачів, аудиторії, формування графу для кожної групи
3. Формування рішень-кандидатів
4. Оцінка рішень-кандидатів
5. Вибір кращого рішення на ітерації та глобальне оновлення феромонного сліда.

Пункти 3, 4, 5 повторюються поки за N ітерацій не буде знайдено таке $L_{iter-best}$, що

$$F(L_{iter-best}) \geq F(L_{all-time-best}) \tag{2.6}$$

Або поки наступна нерівність виконуватиметься

$$|F(L_{all-time-best}) - U| \leq \sigma \tag{2.7}$$

де $L_{all-time-best}$ — найкраще знайдене рішення.

$F(L_{all-time-best})$— оцінка найкращого знайденого рішення.

$L_{iter-best}$ — найкраще рішення, знайдене на ітерації.

$F(L_{iter-best})$— оцінка найкращого рішення, знайденого на ітерації.

U— найбільша можлива оцінка (оцінка зверху).

2.9 Оцінка якості побудованого рішення

Якість рішення-кандидата визначається кількістю не обов'язкових обмежень, яким він задовільняє. Таким чином, оцінка якості k-того рішення-кандидата вираховується наступним чином:

$$F(R_k) = \sum_{i,j \in R_k} m(s_i, t_j)$$

$$m(s_i, t_j) = \sum_{h=0}^{|Q|} c_h * \delta_h$$

де $m_(s_i, t_j)$ — функція оцінки розміщення і-го предмета в j-му таймслоті.

δ_h— функція-індикатор,яка показує чи виконується h-ий критерій.

c_h— ваговий коефіцієнт h - го критерія.

Якщо рішення-кандидидат порушує обов'язкові умови, то його рейтинг дорінює нулю. Такі рішення не будуть враховуватись на наступних ітераціях алгоритма.

Якість рішення кандидата впливає на кількість, "залишеного" при переході від однієї вершини до іншої, феромона (феромонного сліда). Це, в свою чергу, впливає на ймовірність вибору виршини іншим агентом. Так, чим вище якість рішення-кандидата, тим більше воно буде враховуватись іншими агентами на наступних ітераціях.

2.10 Евристична інформація

Навідміну від задачі комівояжера, з якою порівнювалась задача складання розкладу, дана задача не має заздалегіть визначеної евристичної інформації, яку можна використати в роботі алгоритма.

У даному випадку, кожен наступний вибір вершини агентом залежить від попереднього. Тому і евристичну інформацію слід оновлювати кожного разу перед вибором кожної наступної вершини для кожного агента окремо.

Оцінка доцільності вибору наступної вершини включає наступні критерії

– побажання викладачів про час проведення занять

- розміщення ”вікон” на першій, або останній парі.
- розміщення лекцій та практичних занять по одному предмету в один день
- розміщення не більше однієї лекції чи практичного заняття в один робочий день
- розміщення лекцій в першій половині робочого дня, а приктичних та лабораторних занять — в другій
- розміщення лекцій перед практичними заняттями

Для кожної вершини оцінка по цим критеріям сумується, з врахуванням таких самих вагових коефіцієнтів, що і оцінка повного рішення-кандидата. Усі критерії, окрім останнього, потребують перегляду вершин, що були вибрані для одного поточного робочого дня, тобто перегляду 1-5 вершин. Тому такий спосіб врахування евристичної інформації є достатньо ефективним.

2.11 Висновки до розділу 2

Загалом, алгоритми, що застосовуються для вирішення NP - повних задач поділяються на точні алгоритми - гарантовано знаходять найбільш оптимальний розв’язок, проте їх складність, зі збільшенням об’єму даних, росте за експоненціальним законом та наближені алгоритми - знаходять розв’язок задачі, наближений до найбільш оптимального, проте є ефективними за часом роботи. Перші не доцільно використовувати в прикладних задачах. З огляду на дослідження ефективності евристичних алгоритмів для задачі комівояжера базисним для задачі складання розкладу обрано алгоритм систем мурашиних колоній.

Задачу складання розкладу доцільно представити у вигляді графа, що складається окремо для кожної навчальної групи. Після чого, запуситити ітераційний процес модифікованого алгоритму систем мурашиних колоній. Використовуючи функцію оцінки, пріорітезувати найкращі знайдені рішення-кандидати.

3 ПРАКТИЧНЕ ЗАСТОСУВАННЯ

3.1 Деталі реалізації

3.1.1 Версії використаних бібліотек та фреймворків

Для реалізації програмного продукта на мові C++ з підтримкою стандарту c++20 було використано компілятор g++ (Pop!OS 20200416) 10.2.0[master revision 3c3f12e2a76:dcee354ce56:44b326839d864fc10c459916abcc97f35a9ac3de]. Фреймворк Qt версії 5.14.0. Графіки було побудовано за допомогою бібліотеки matplotlibcpp.

3.1.2 Опис використаних бібліотек та фреймворків

Головним вікном програми є об’єкт класу MainWindow який спадкується від стандартного класу бібліотеки Qt QMainWindow. Для реалізації меню програми, що розміщене ліворуч від головної робочої області було використано плагін SideBar. Його елементами є об’єкти класу QAction. Всі таблиці є об’єктами класу QTableWidget.Також деякі екрани містять декілька таблиць, що розміщені в табах віджета QTabWidget. Також для функціоналу кнопок використані QPushButton. Повідомлення про помилку відображаються у окремому вікні за допомогою QMessageBox.

Екран з таблицями для відображення розкладу, відрізняються від інших таблиць. Для його побудови було використано QGridLayout та QWidget, всередині якого знахояться QLabel для відображення текстової інформації. Всі графічні ефекти було реалізовано з використанням класу QPropertyAnimation.

Форма, яка була створена в QtDesigner не є компонентом головної програми,оскільки мова C++ не підтримує створення форм за допомогою мови розмітки XML. Відповідно до форми генерується код на мовах C++ або Python, як показано на рисунку 3.25.

Для навігації за допомогою елементів меню використовується підхід сигналів і слотів, який є реалізацією поведінкового шаблона проектування Observer. Його основний принцип полягає в тому, що компонент може посилати сигнали, які зберігають в собі інформацію про деяку подію (наприклад натискання

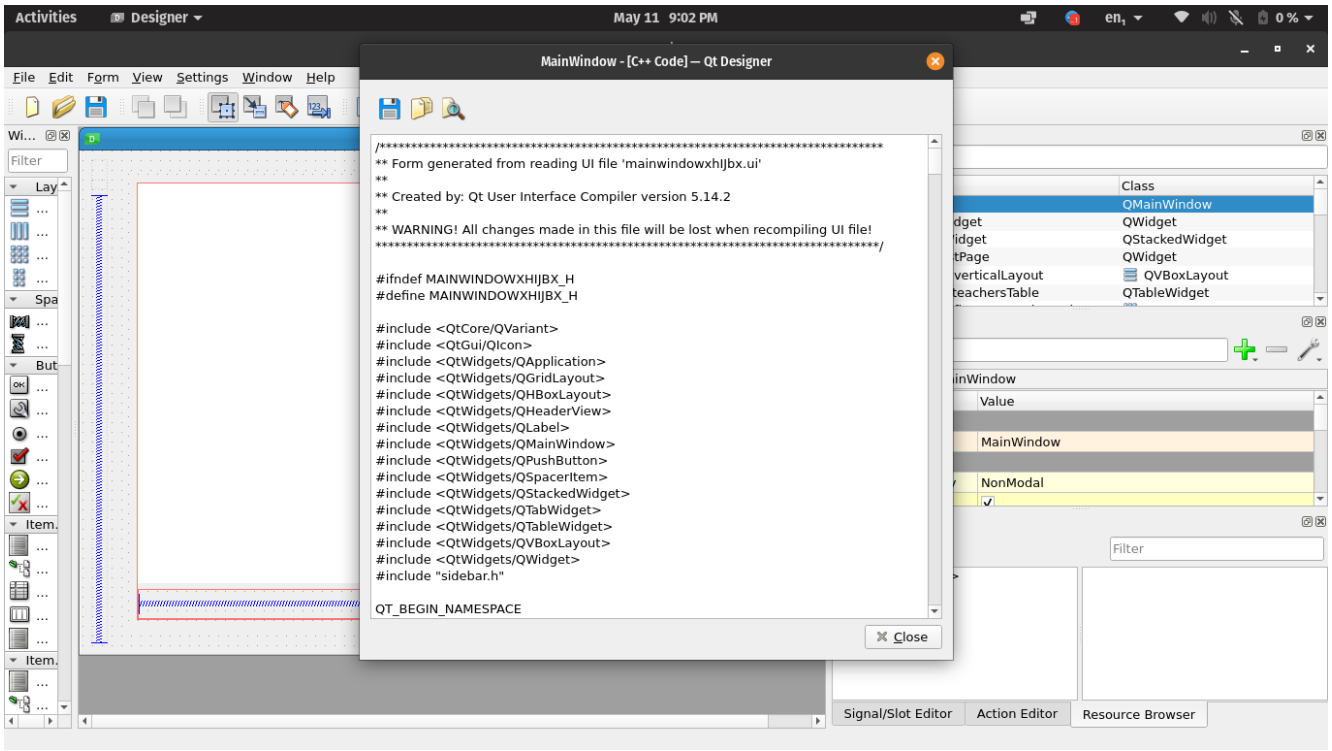

Рисунок 3.1 – Приклад згенерованого коду на мові C++ за допомогою QtDesinger

кнопки, змінення тексту, тощо). А компонент, який очікує на сигнал, при його отриманні виконує задалегіть визначені дії.

3.2 Опис програмного продукту

Для реалізації алгоритмічної частини програмного продукту було використано мову програмування C++. Такий вибір було зроблено з огляду на те, що в пріорітеті стоїть швидкість та надійність роботи програмного продукту. Обрана мова програмування є однією з найшвидших, та забезпечує безперебійну роботу додатків завдяки гнучкому механізму обробки виключень. C++ є кросплатформеною мовою програмування, це дає змогу запускати додатки чи використовувати бібліотеки на різних платформах та пристроях. Це дає користувачеві більше можливостей при виборі девайсів та операційних систем.

Для реалізації графічного інтерфейса користувача було використано фреймворк Qt. Для автоматизованого створення форм — допоміжне програмне забезпечення Qt Desinger. Приклад дизайну форми з використанням цієї програми наведено на рисунку нижче.

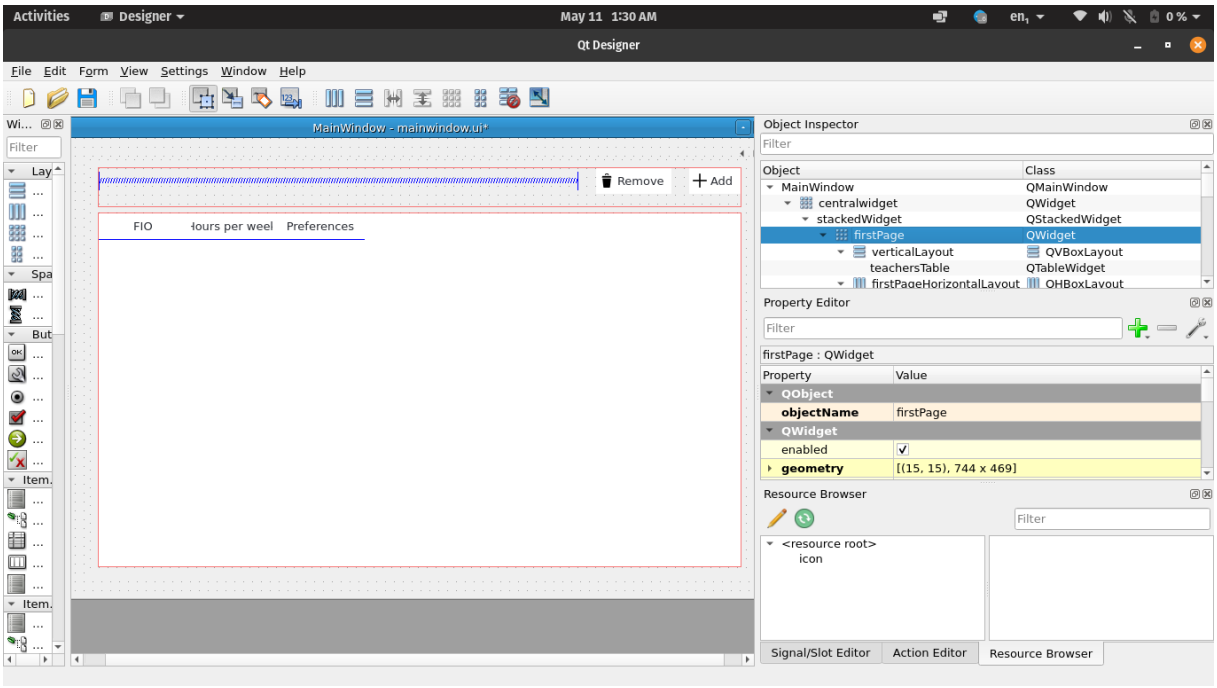

Рисунок 3.2 – Приклад використання Qt Desinger для програмного продукту дипломної роботи

Почтакове вікно створеного в дипломній роботі програмного продукту виглядає наступним чином:

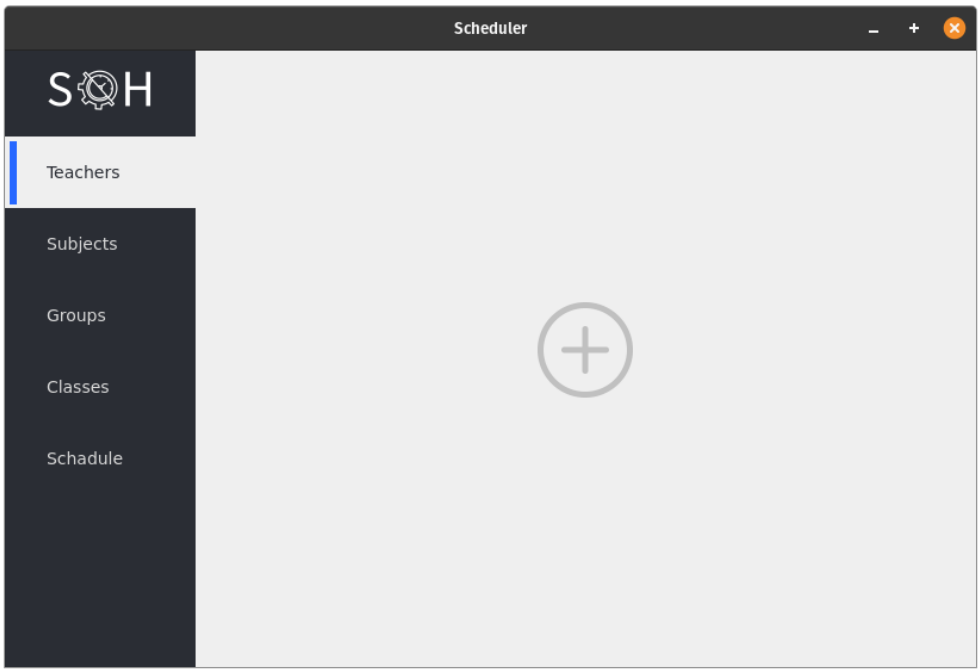

Рисунок 3.3 – Початкове вікно програмного продукту

Програмний продукт для автоматизовного складання розкладу в навчальних закладах має наступні функції:

1. Додати викладача
2. Додати предмет
3. Додати групу та граф для неї
4. Додати аудиторію
5. Створити та редагувати розклад

Кожен з перелічених елементів функціоналу підтримує CRUD операції. Далі буде розглянуто детально кожну функцію, відповідний графічний інтерфейс та послідовність дій для її використання.

3.3 Екран додавання інформації про викладача

Для початку роботи та створення таблиці з інформацією про викладачів необхідно натиснути на довільну область віджета зі знаком плюс. Його загальний вигляд зображено на рисунку нижче.

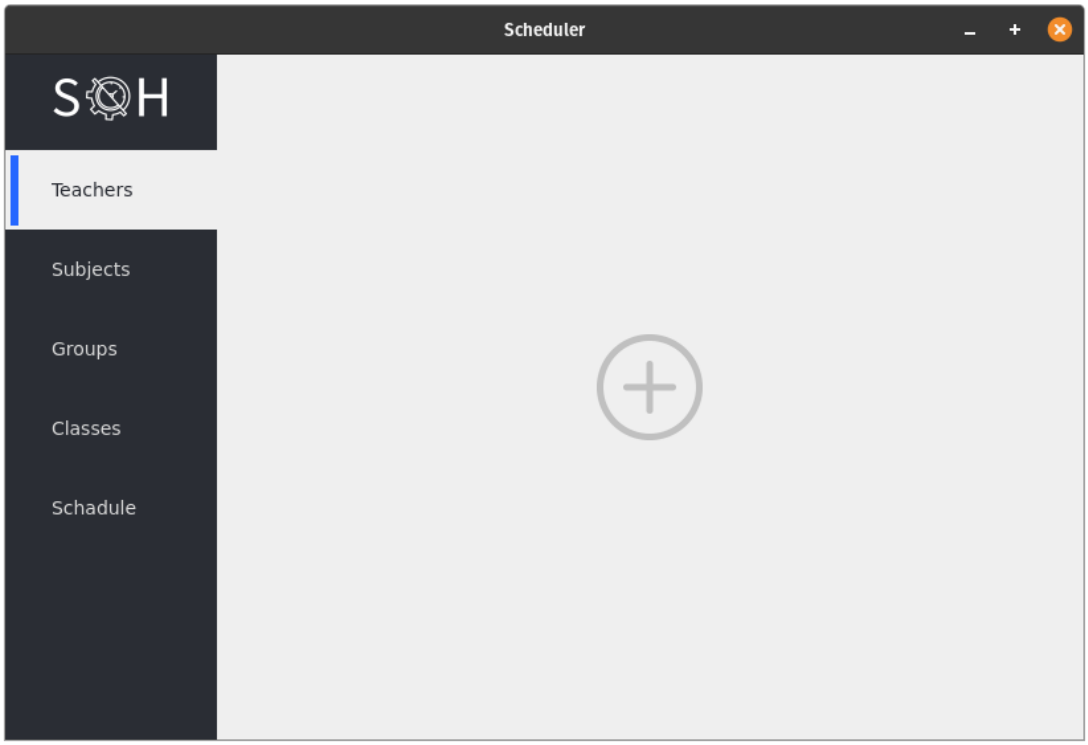

Рисунок 3.4 – Початковий стан екрану функції додавання викладача

Після цього буде відображено порожню таблицю. Додати інформацію про викладача можна натиснувши кнопку *Add item.*. Результат виконання заображено на рис.3.4. Було відображено нову строку в таблиці, де можна внести інформацію про П.І.Б. викладача, кількість робочих академічних годин за тиждень та вподобання про час проведення занять. Приділимо особливу увагу останній опції. Якщо викладачу зручно проводити заняття в будь-який час або відсутня інформація про його вподобання необхідно обрати опцію *Any time*. Для вибору певних робочих годин, які є більш бажаними необхідно обрати опцію *Set manualy*.Результат виконання зображено на рис.3.5.

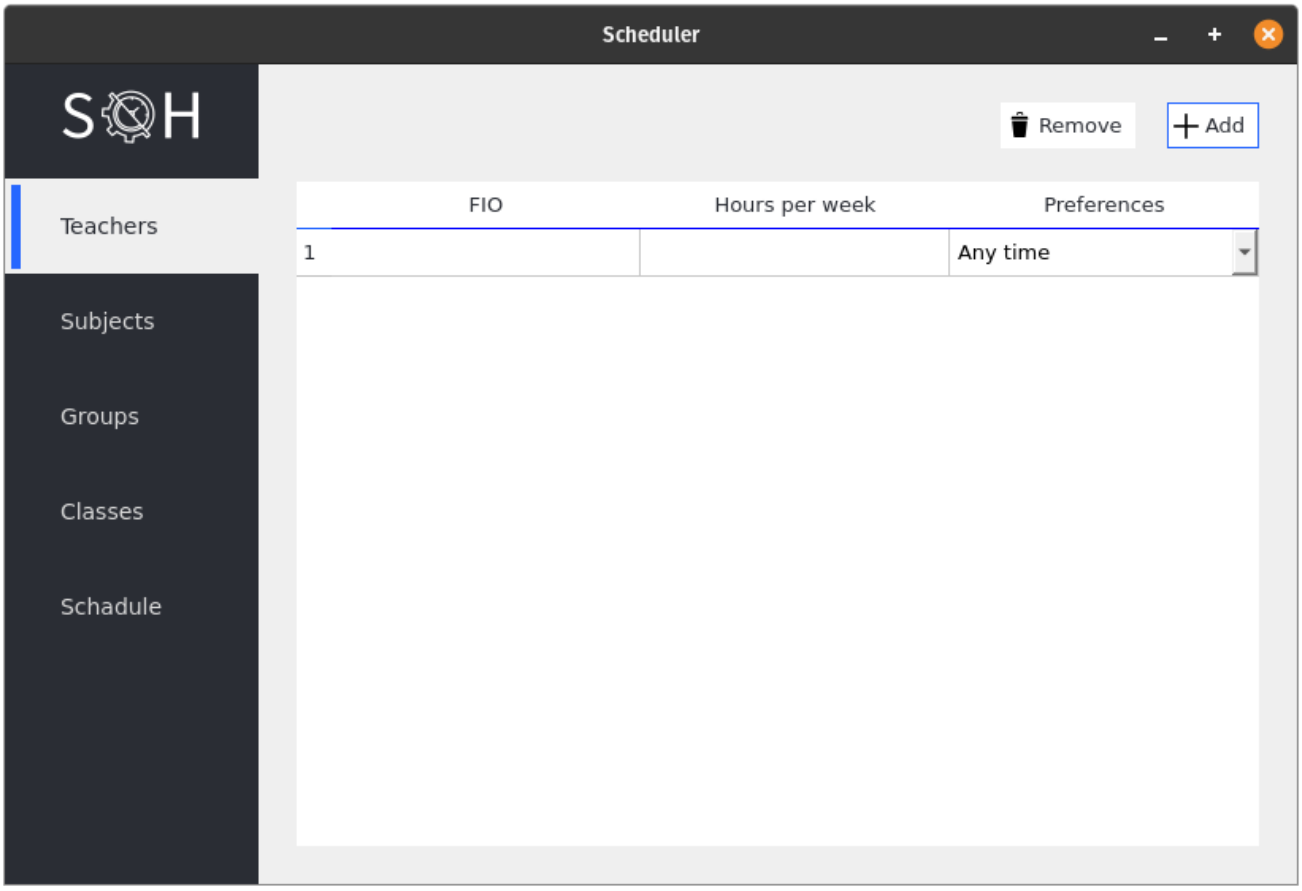

Рисунок 3.5 – Екран додавання інформації про викладача

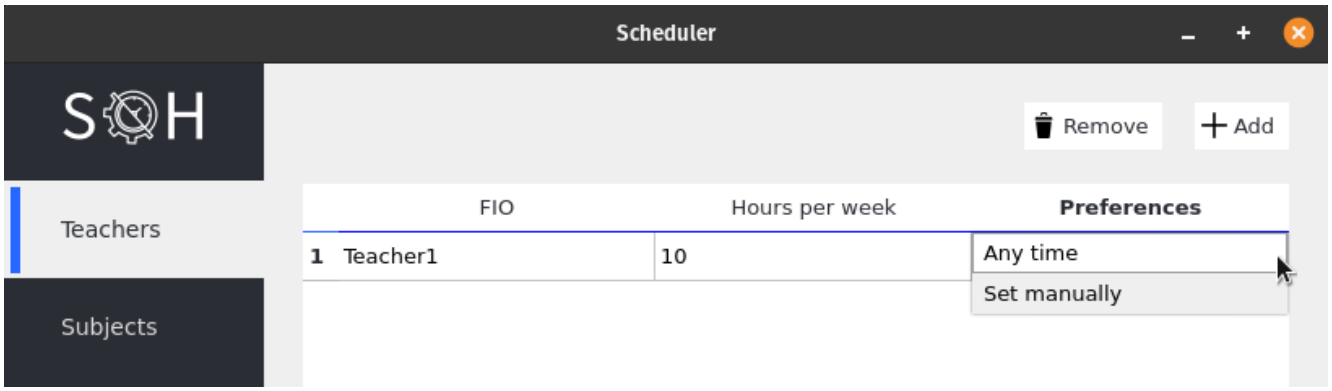

Рисунок 3.6 – Початковий стан екрану функції додавання викладача

При виборі останньої буде створено новий віджет, як показано на рисунку 3.6 нижче. Після внесення необхідної інформації потрібно натиснути на кнопку *Save*.

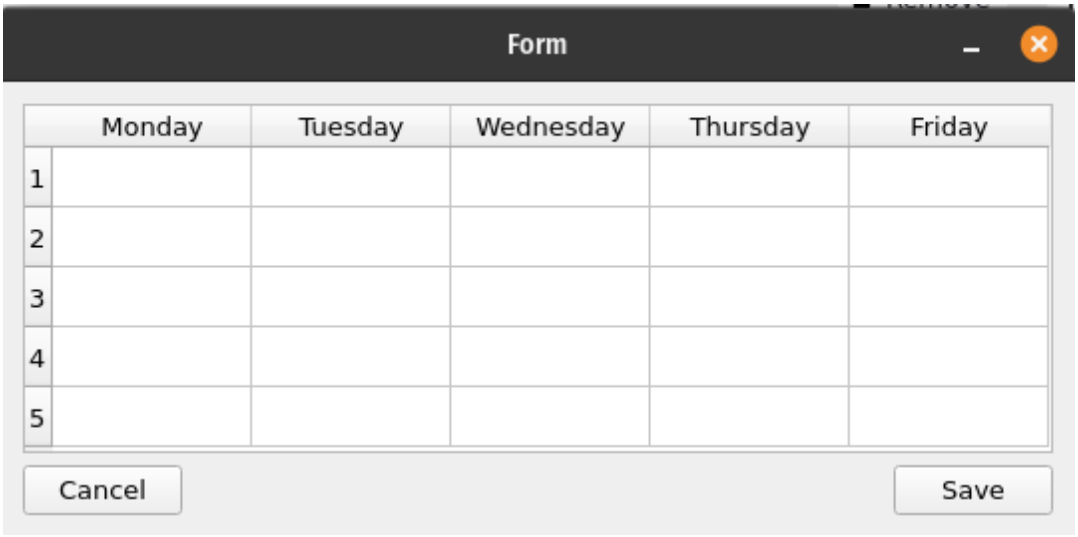

Рисунок 3.7 – Віджет для внесення інформації про вподобання викладачів про час проведення занять

Видалити інформацію про викладача можна натиснувши кнопку *Remove item*. Фреймворк Qt забезпечує широкі можливості для обробки виключних ситуацій. Так, наприклад, при спробі видалення елемента з порожньої таблиці нічого відбуватись не буде. При спробі додавання нової строки, без коректоного заповнення попередньої, буде отримане наступне повідомлення виключення:

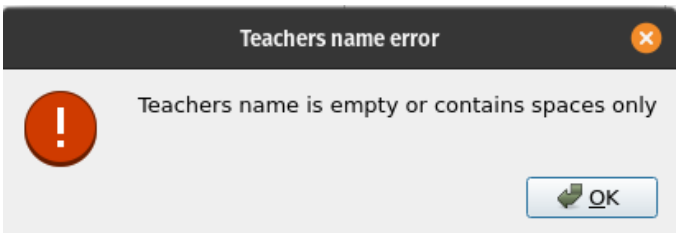

Рисунок 3.8 – Приклад обробки виключних ситуацій

Така обробка виключень наявна на всіх екранах, що будуть описані нижче.

3.4 Додавання інформації про предмет

Для початку роботи та створення таблиці з інформацією про навчальний предмет необхідно натиснути на довільну область віджета зі знаком плюс. Його загальний вигляд зображено на рисунку нижче.

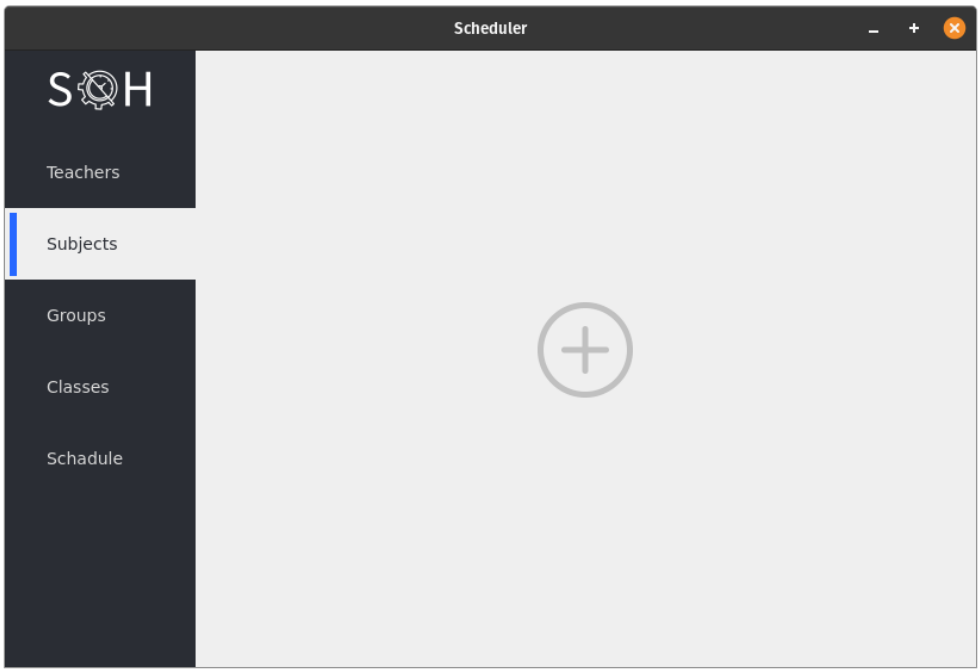

Рисунок 3.9 – Початковий стан екрану функції додавання інформації про навчальний предмет

Після цього буде відображено порожню таблицю. Додати інформацію про навчальний предмет можна натиснувши кнопку *Add*.

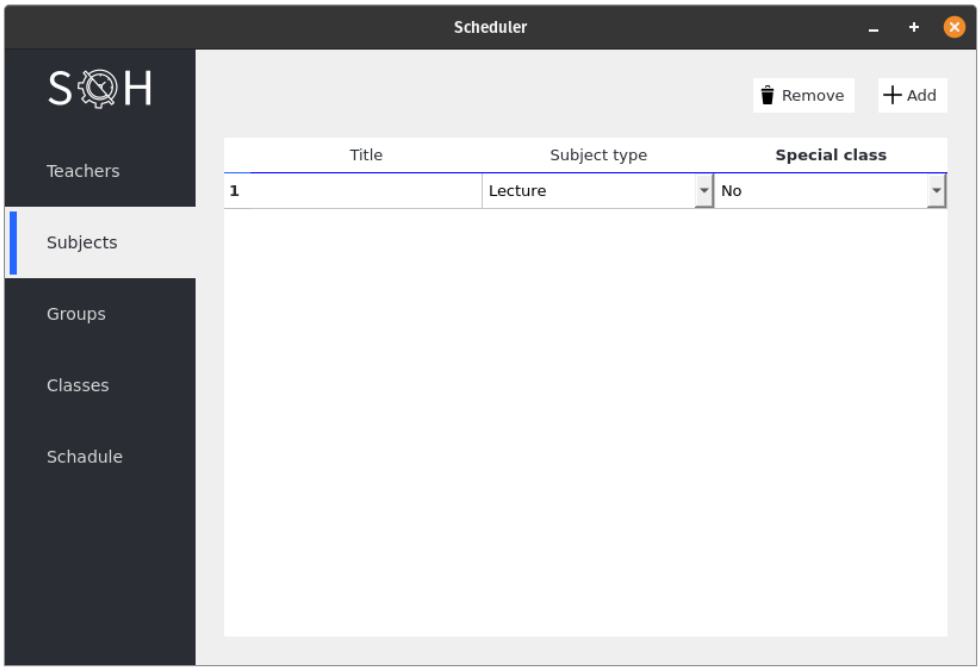

Рисунок 3.10 – Екран додавання інформації про викладача

Було відображено нову строку в таблиці, де можна внести інформацію про назву предмета, його тип та необхідність додаткового обладання для проведення заняття. Приклад вибору типа предмета зображено на рисунку нижче.

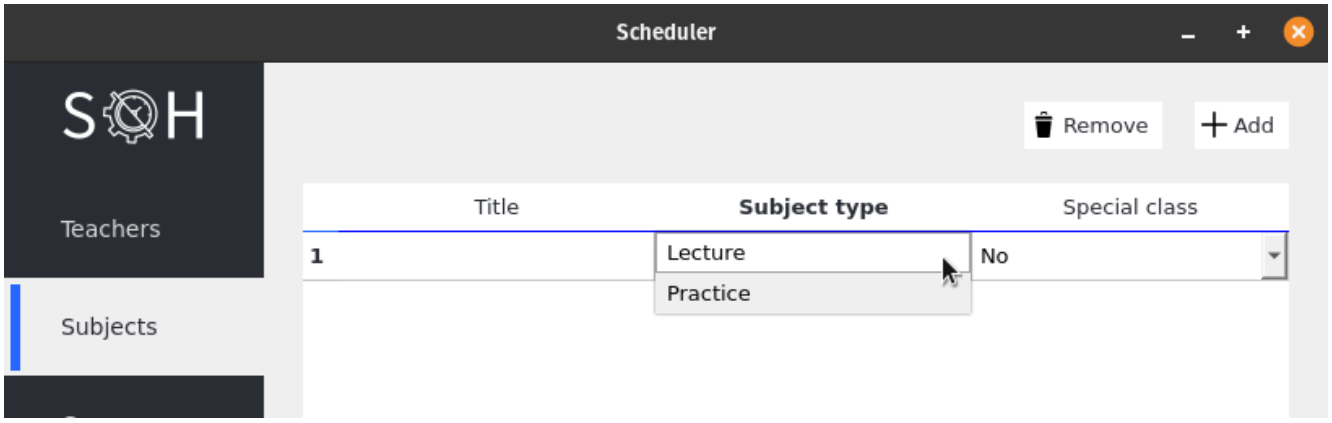

Рисунок 3.11 – Приклад вибору типа навчального предмета

Приклад вибору необхідності додаткового обладання для проведення заняття зображено на рисунку 3.12.

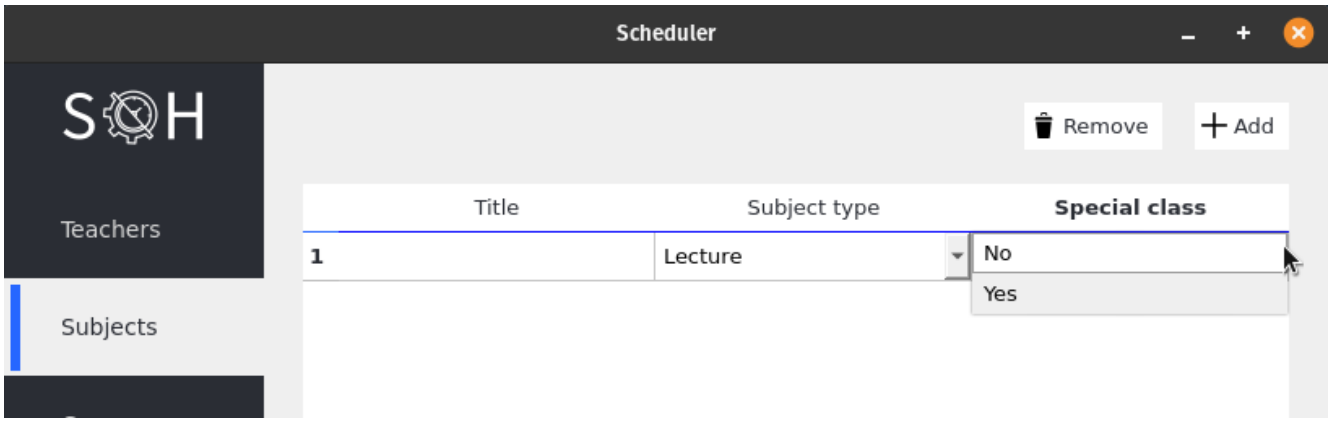

Рисунок 3.12 – Приклад вибору необхідності додаткового обладання для проведення заняття

3.5 Додавання інформації про групу

Для початку роботи та створення таблиці з інформацією про навчальну групу необхідно натиснути на довільну область віджета зі знаком плюс. Його загальний вигляд зображено на рисунку нижче.

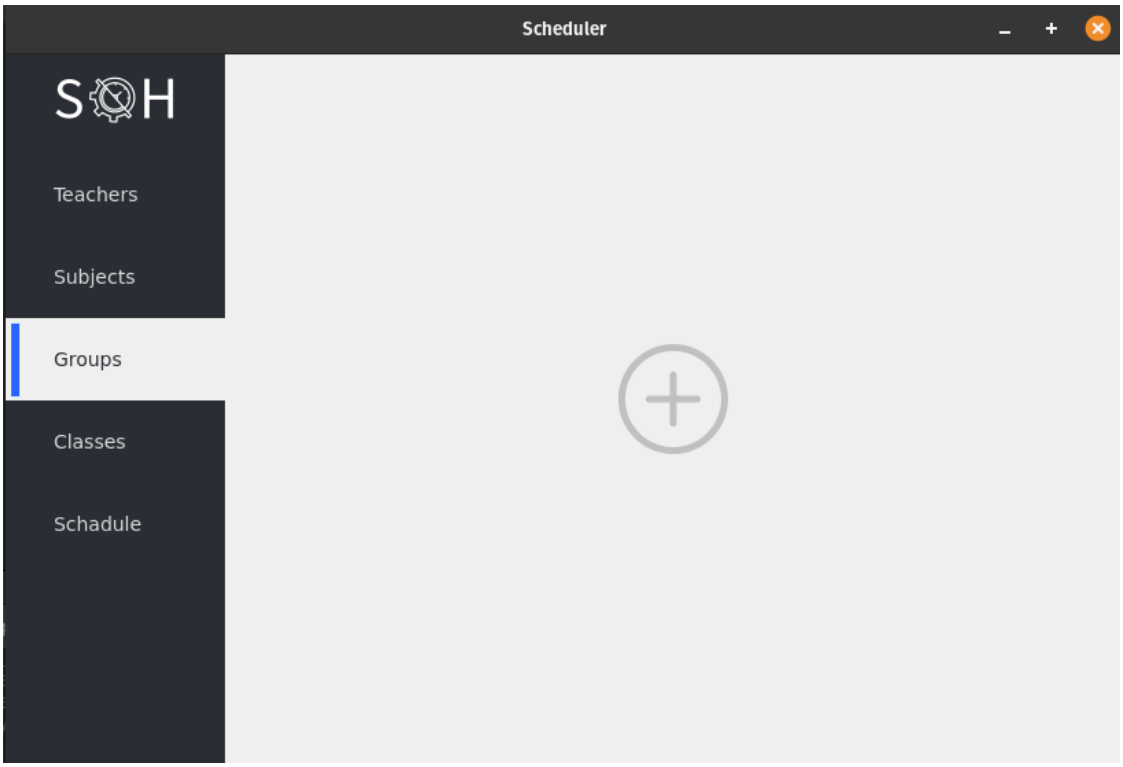

Рисунок 3.13 – Початковий стан екрану функції додавання інформації про навчальну групу

Після цього буде відображено порожню таблицю. Додати інформацію про навчальну групу можна натиснувши кнопку *Add group*. Після цього відкриється вікно, в строку якого можна ввести назву групи.

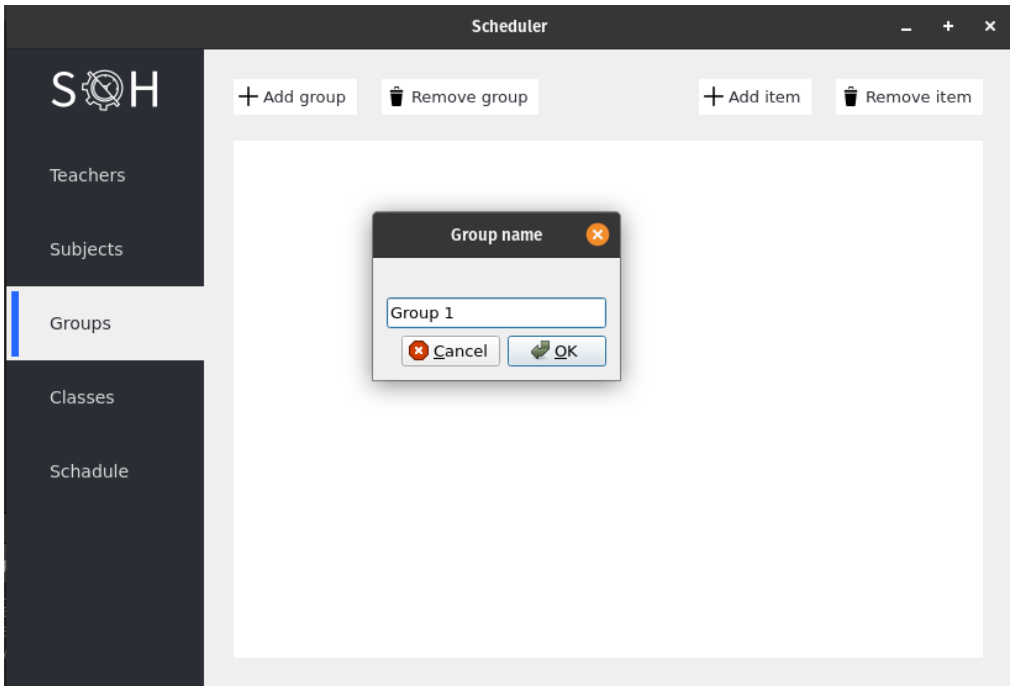

Рисунок 3.14 – Екран додавання назви групи

Після натискання кнопки *Ok*, буде відображена порожня таблиця, де можна додати пару виду предмет та викладач та задати кількість їх годин для групи. Це є процесом формування графу для однієї групи.

Викладача можна вибрати з переліку викладачів, які було наведено у вікні додавання викладача. Приклад такого вибору зображено на рисунку нижче.

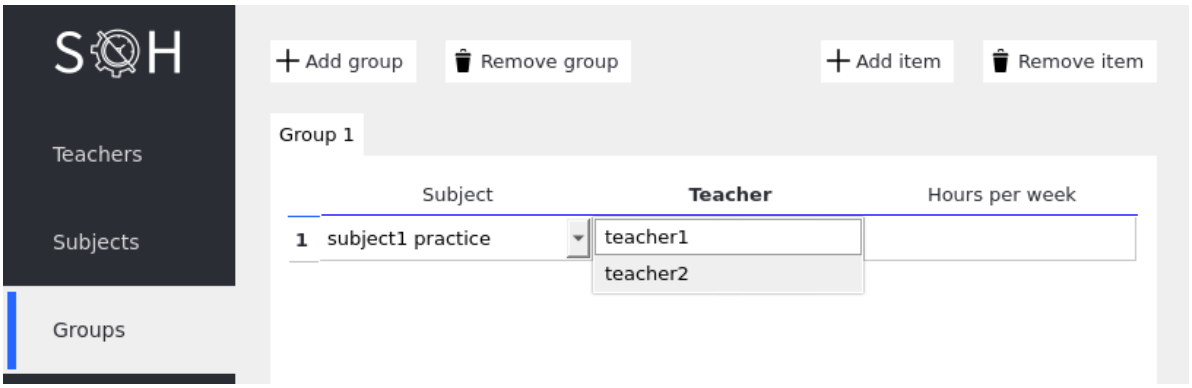

Рисунок 3.16 – Приклад вибору навчального предмета

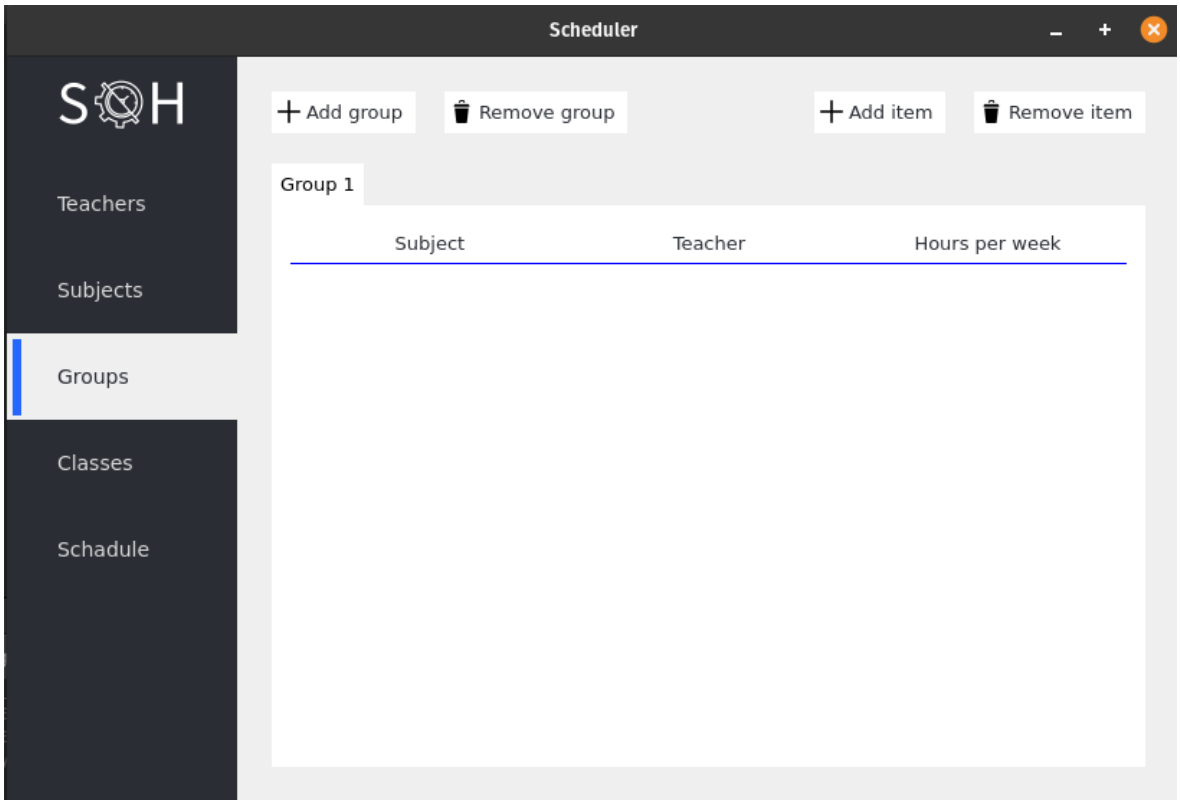

Рисунок 3.15 – Таблиця для внесення інформаціх про групу

Навчальний предмет можна вибрати з переліку предметів, які було наведено у вікні додавання предмета. Приклад такого вибору зображено на рисунку нижче.

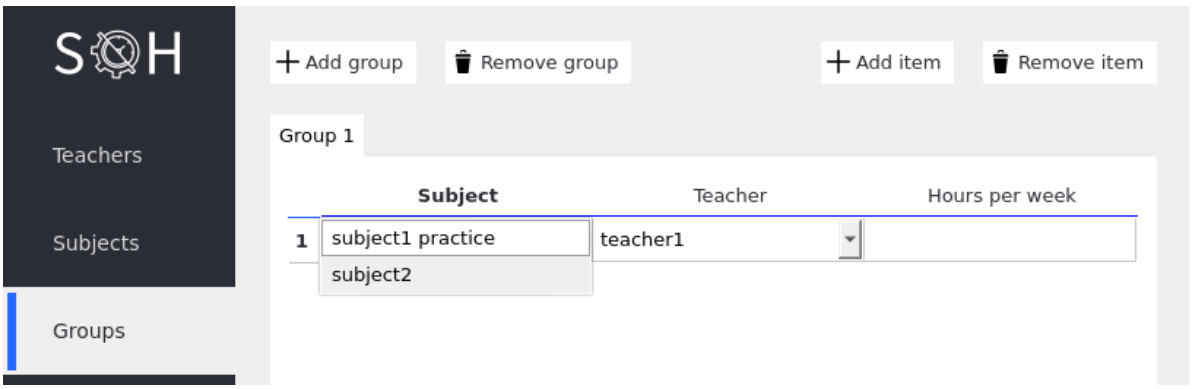

Рисунок 3.17 – Приклад вибору навчального предмета

Механізм виключень аналогічний до вікон, що описані в розділах вище.

3.6 Додавання інформації про аудиторії

Для початку роботи та створення таблиці з інформацією про навчальні аудиторії необхідно натиснути на довільну область віджета зі знаком плюс.

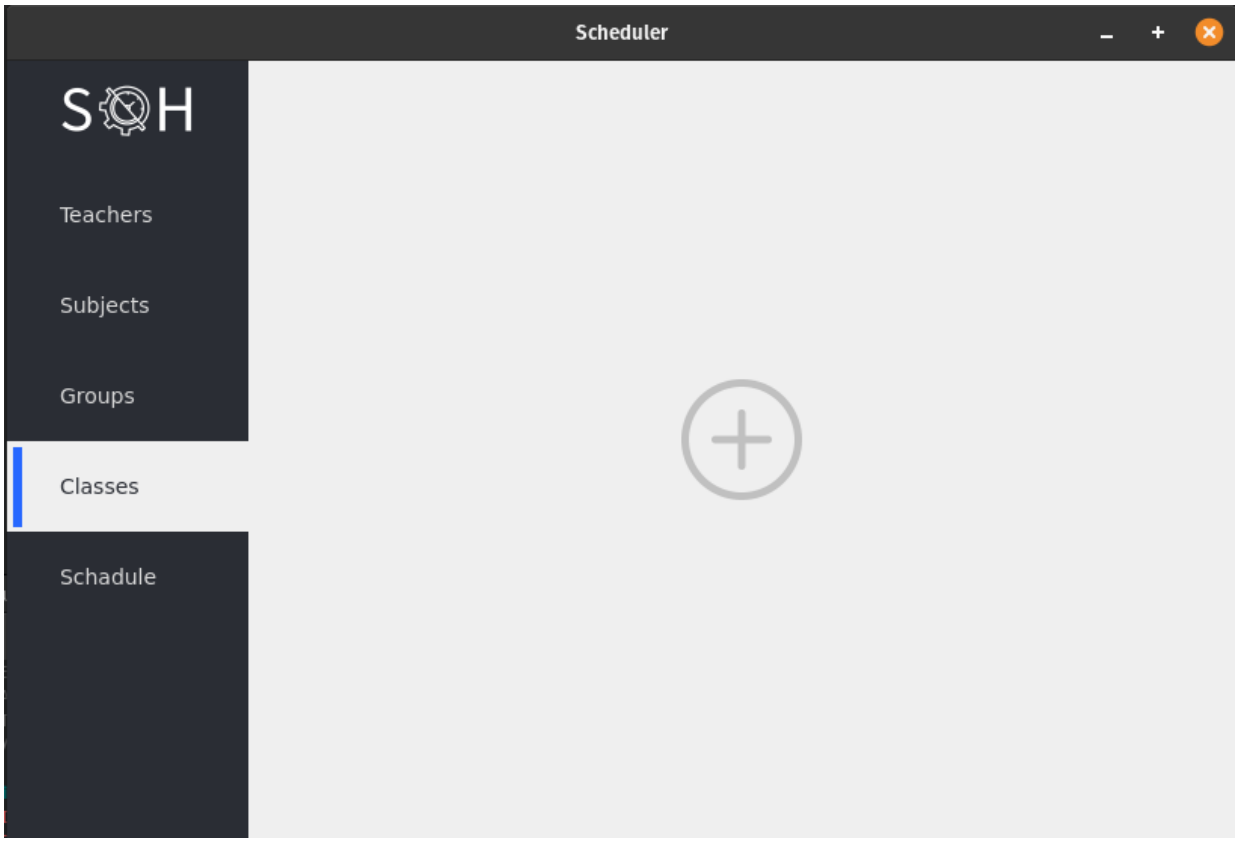

Рисунок 3.18 – Початковий стан екрану функції додавання інформації про навчальні аудиторії

Результат виконання зображено на рис. 3.17. Після цього буде відображено порожню таблицю. Додати інформацію про навчальний предмет можна натиснувши кнопку *Add item*. Було відображено нову строку в таблиці, де можна внести інформацію про номер аудиторії,номер корпусу, де розміщена аудиторія, місткість аудиторії та наявність додаткового обладнення. Приклад вибору типу аудиторі(за останнім, з перелічених, критеріїв) зображено на рисунку нижче.

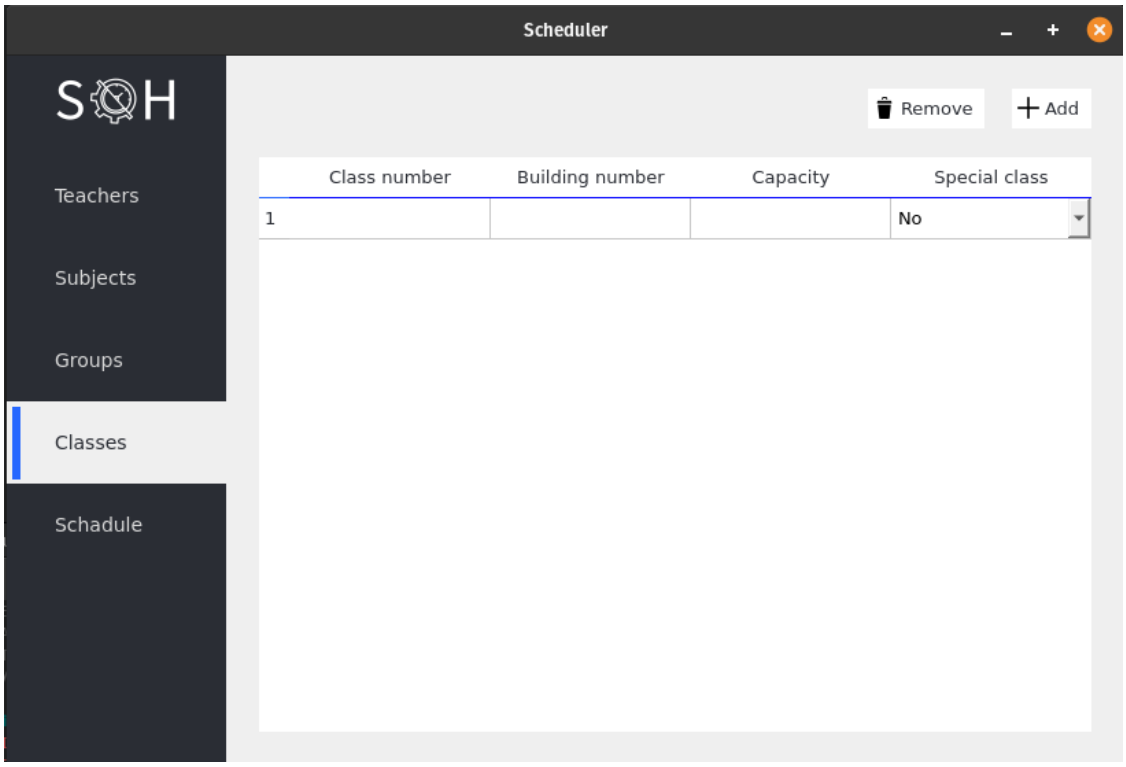

Рисунок 3.19 – Екран додавання інформації про навчальну аудиторію

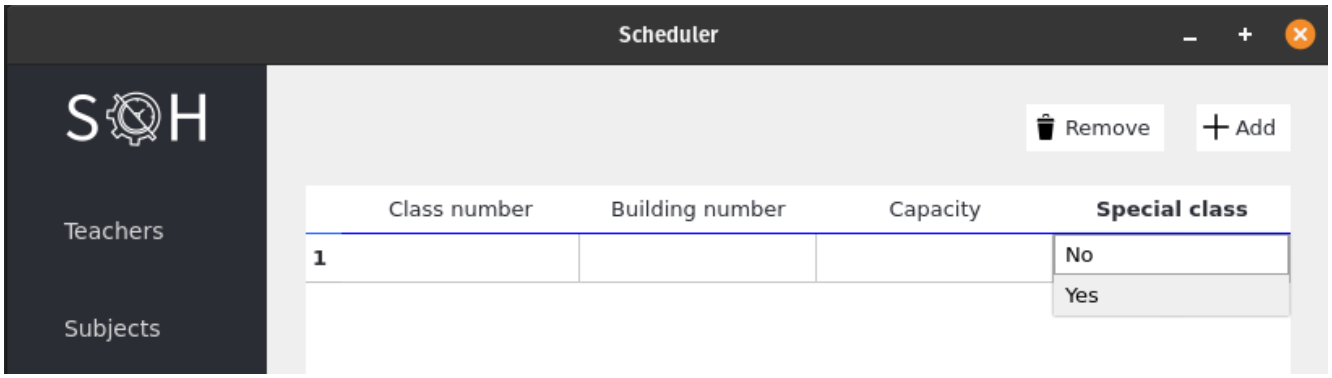

Рисунок 3.20 – Приклад вибору типу навчальної аудиторії

3.7 Створення розкладу

Для початку роботи алгоритма стоврення розкладу, необхідно натиснути на кнопку *Create*. Початковий екран функції створення розкладу зображено на рисунку нижче.

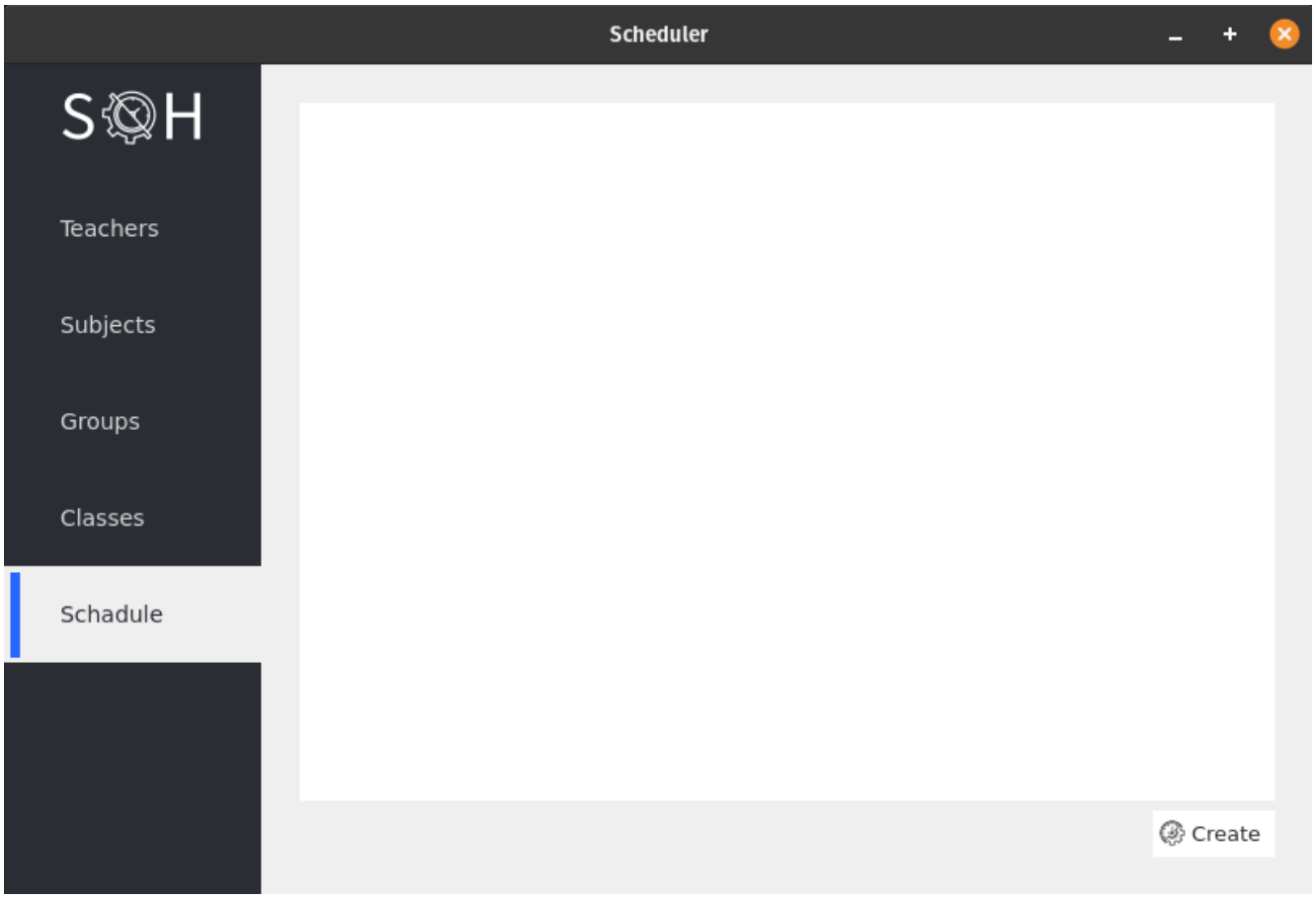

Рисунок 3.21 – Почтаковий екран функції створення розкладу

Після цього, буде зчитано інформацію зі всіх таблиць, що були описані у попередніх розділах(таблиці з інформацією про викладачів, навчальні предмети, навчальні групи та навчальні аудиторії) та передано на вхід алгоритма для складння розкладу. У результаті буде відображено віджет, що містить дві функціональні таблиці зі створеним розкладом для навчальних груп (студентів) та кожного з викладачів. Приклад інтерактивної таблиці з розкладом для студентських навчальних груп наведено на рисунку 3.22, а викладацький розклад на рисунку 3.23.

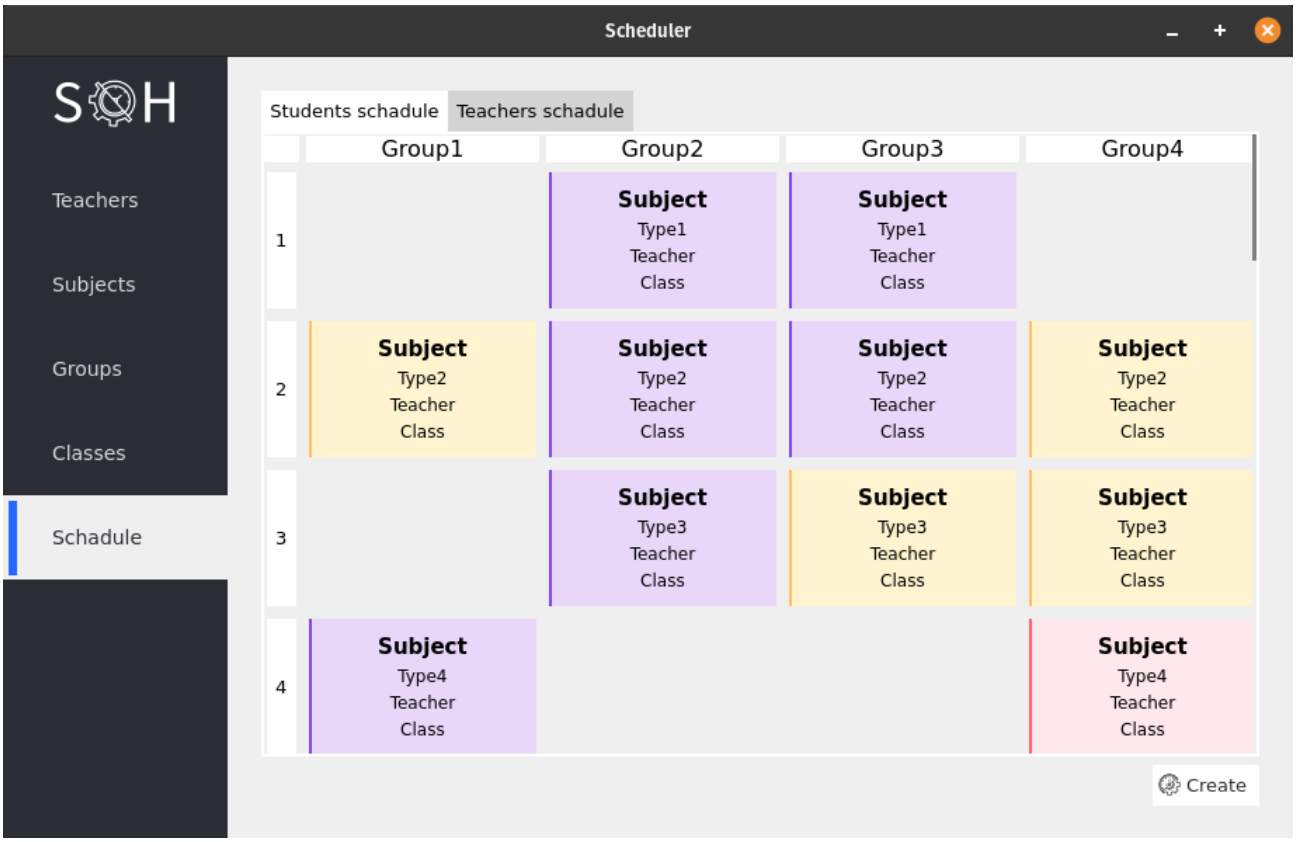

Рисунок 3.22 – Приклад розкладу для студентів

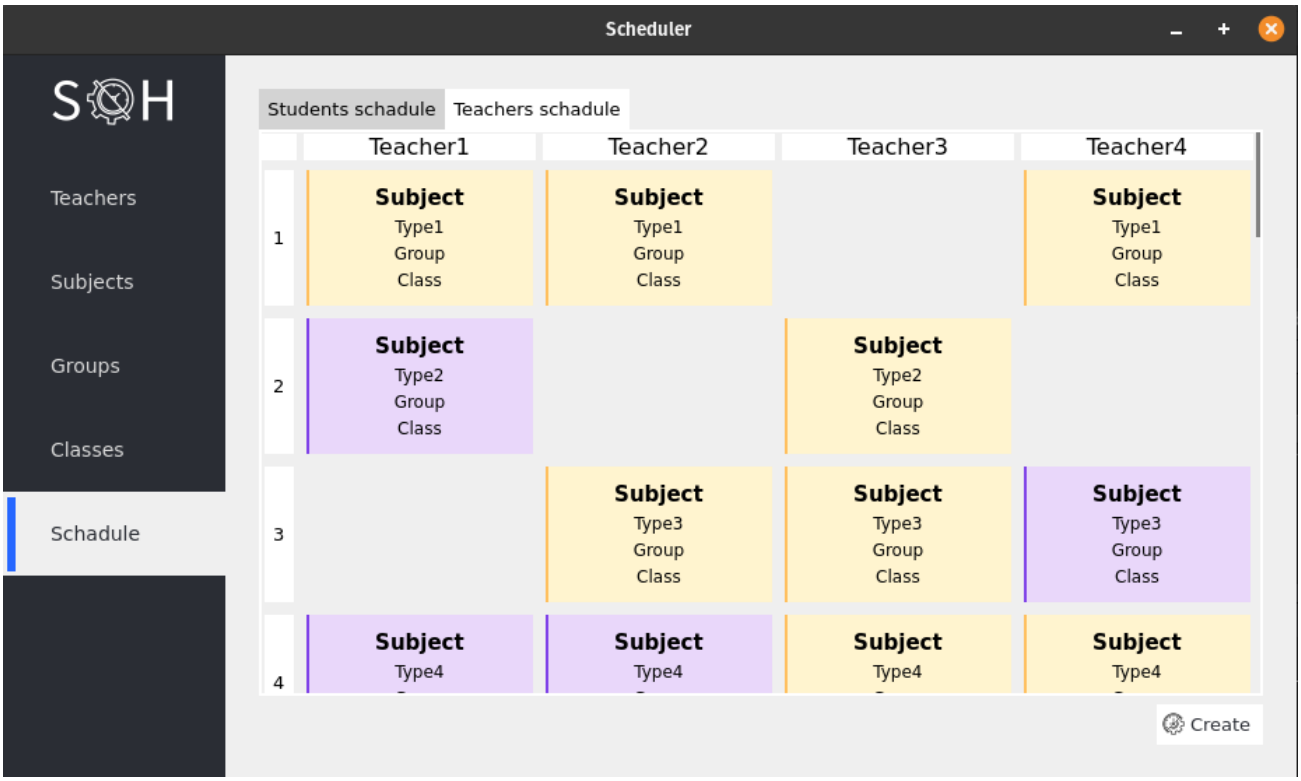

Рисунок 3.23 – Приклад розкладу для викладачів

Віджети, на яких відображено елементи розкладу, є функціональними. Зокрема, наявна можливість міняти місцями комірки розкладу шляхом послідовного кліка правою кнопкою миші на 2 віджети. Приклад зміни показано на рисунку нижче.

Однак, якщо надати користувачу можливість міняти місцями будь-які два елементи розкладу, це може призвести до великої кількості протирічь в розкладі або навість зробити його недійсним. Тому існують два правила, які обмежують цей функціонал.

1. Можна міняти місцями елементи в рамках розкладу одного викладача, заздалегіть перевіривши, чи не виникає протирічь при такій зміні в розкладі груп, для яких також відбувається заміна.
2. Можна міняти місцями елементи в рамках розкладу однієї групи, заздалегіть перевіривши, чи не виникає протирічь при такій зміні в розкладі викладачів, для яких також відбувається заміна.

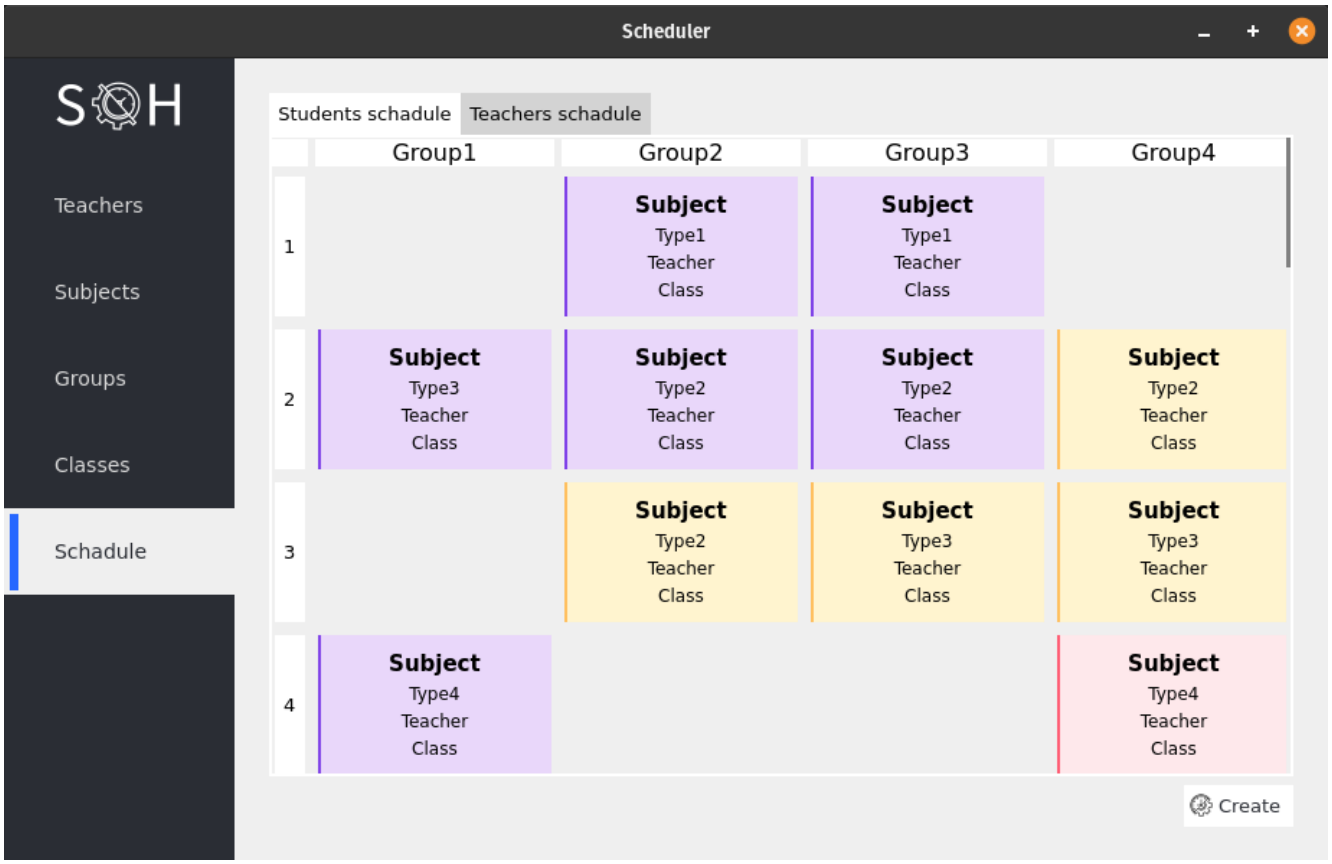

Рисунок 3.24 – Приклад зміни комірок розкладу для студентів

В обох випадках, якщо протиріччя виникають зміну провести не можливо. Приклад роботи зобрадено на рис. 3.23.

Наявна можливість збільшувати розмір комірки шляхом кліка лівою кнопкою миші. Приклад показано на рисунку нижче.

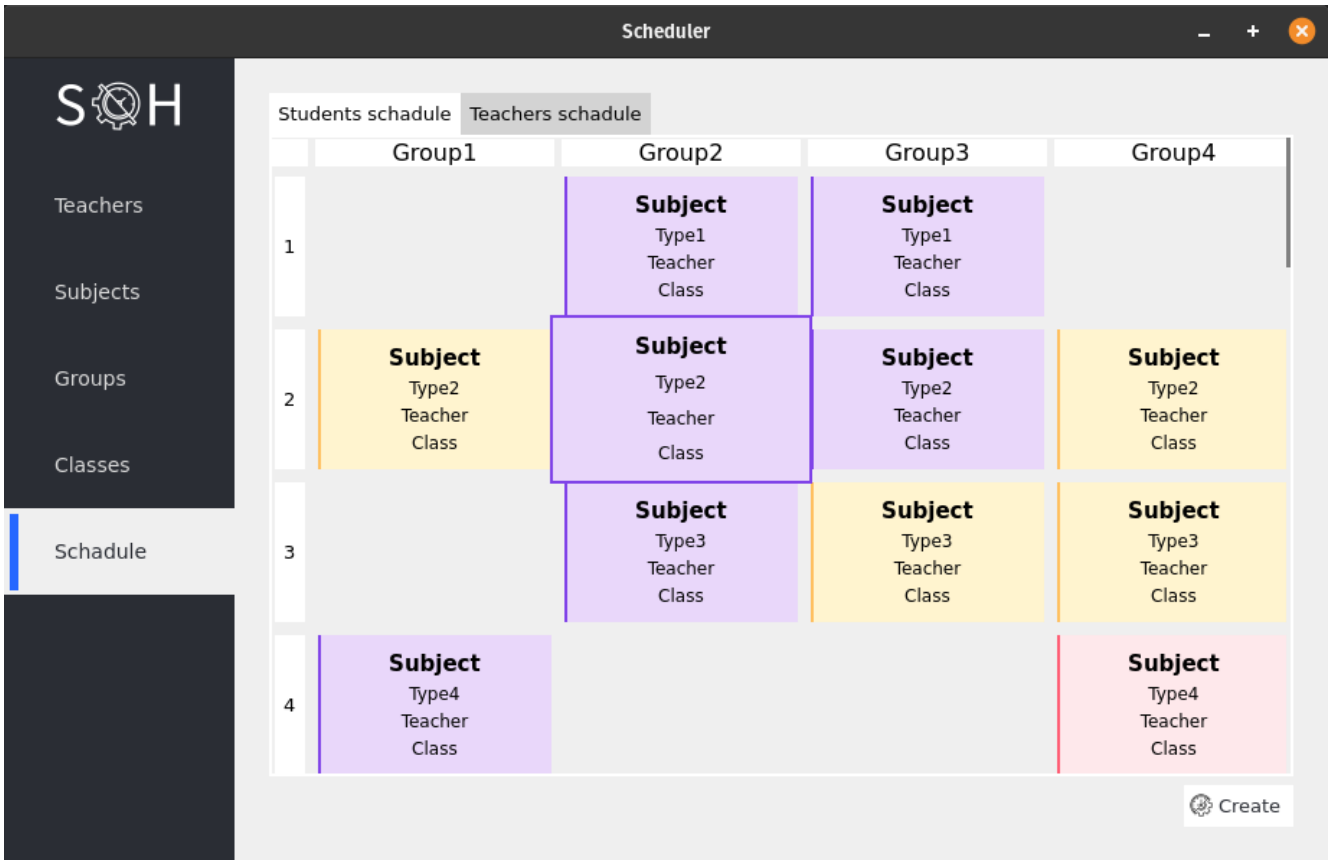

Рисунок 3.25 – Приклад використання функції збільшення комірки розкладу для студентів

Останні дві функції можна використовувати аналогічним чином і з таблицею розкладу для викладачів.

3.8 Аналіз ефективності роботи запропонованого для автоматизованого складання розкладу алгоритма

3.8.1 Ефективність роботи запропонованого для автоматизованого складання розкладу алгоритма

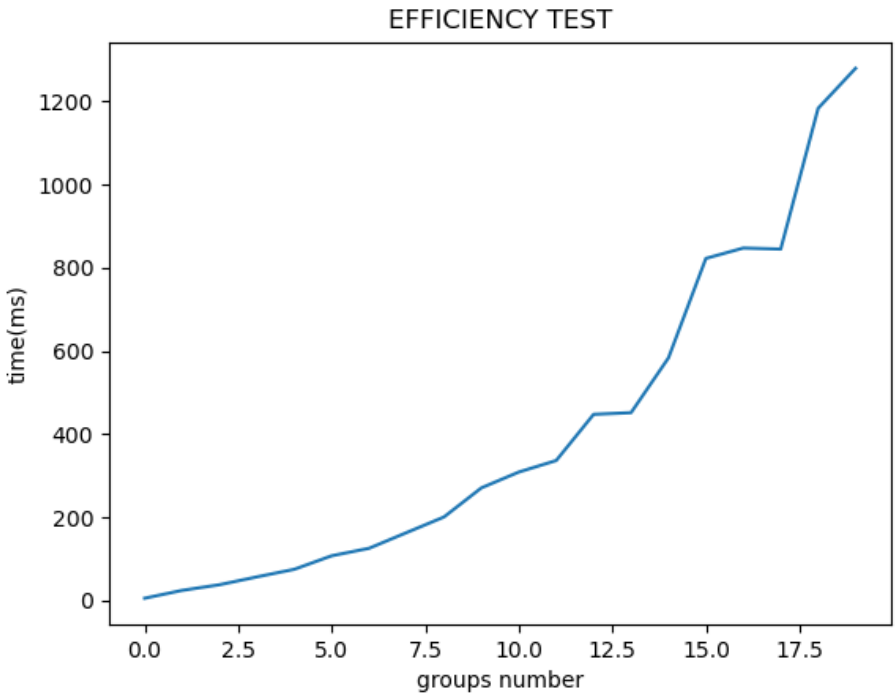

Рисунок 3.26 – Графік залежності часу роботи від кількості груп для ACS

Тестування ефективності роботи алгоритма проводилось на вибірці з n груп, де $n \in [1, 20]$. Для кожної розмірності було згенеровано по 50 випадкових вибірок з множинами викладачів, студентів та аудиторій. Також випадковим чином було згенеровано граф для кожної групи та побажання викладачів про час проведення заняття. На рисунку 3.26 зображено залежність часу роботи алгоритма від кількості груп , для яких одночасно необхідно скласти розклад.

3.8.2 Порівняння ефективності роботи запропонованого алгоритма з генетичним алгоритмом

Для того, щоб впевнитись в тому, що запропонований в кваліфікаційній роботі алгоритм, який базується на алгоритмі систем мурашиних колоній, є дійсно ефективним у порівнянні з інишми евристичними алгоритмами, було проведено порівняльний тест ефективності з одним із їх найпопулярніших представинків, генетичним алгоритмом. Принцип роботи генетичного алгоритму було описано в розділі 2.

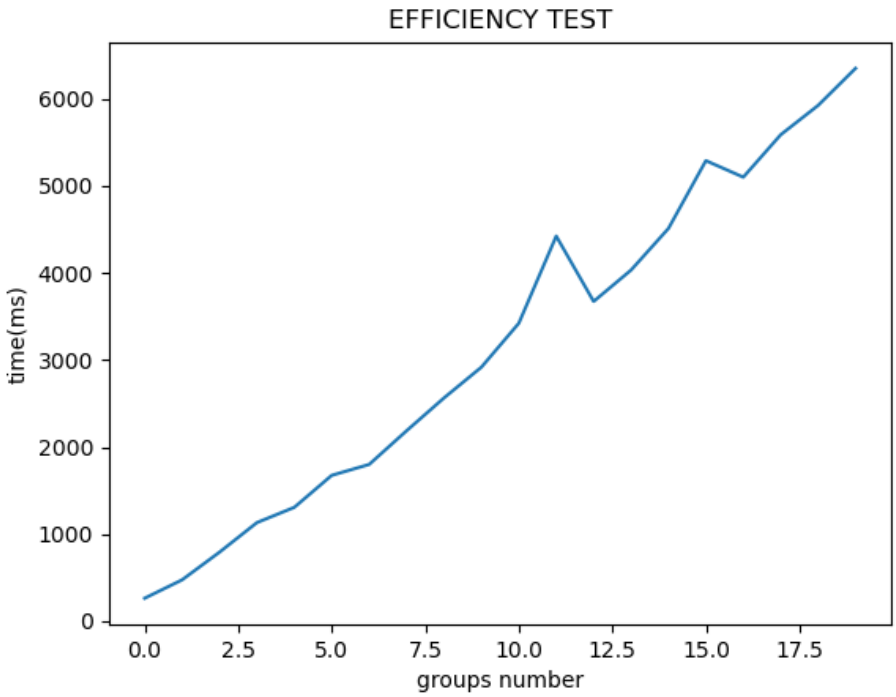

Рисунок 3.27 – Графік залежності часу роботи від кількості груп для генетичного алгоритма

Додатковою задачею було згенерувати початкову популяцію для роботи генетичного алгоритма, а саме m особин, що представляють собою рішенням-кандидати, що не суперечать обов'язковим вимогам, які накладаються на розклад. Для цієї задачі було застосовано ACS з кількістю агентів що дорівнює кількості особин в популяції.

Тестування ефективності роботи алгоритма проводилось на вибірці з n груп, де $n \in [1, 10]$. Час на генерацію популяції не враховувався. На рисунку 3.27 зображено залежність часу роботи алгоритма від кількості груп , для яких одночасно необхідно скласти розклад.

3.8.3 Порівняння ефективності роботи запропонованого алгоритму з алгоритмом повного перебору

Виходячи з того, что задача складання розкладу відноситься до класу NP-повних, метод повного перебору матиме занадто високу, для отримання

парктичних результатів, часову складність. Проте, можемо оцінити час, який буде витрачено на побудову розкладу.

Нехай необхідно скласти розклад для однієї групи, повного робочого тижня(тобто 25 робочих академічних годин). Відповідно, граф буде мати 25 вершин. Кількість варіантів впорядкування 25 вершин дорівнює

$$n_{variants} = 25! = 1.551121e + 25$$

.

Практичним чином було визначено час, який витрачається на складання та оцінку $10'000$ рішень-кандидатів.

$$t_{10} = 7981$$

мс, тоді на обрахунок всіх варіантів буде вистрачено

$$t_{gr} = n_{variants}/10'000 * t_{10} = 1.237949673558246e + 23(c)$$

Це є підтвердженням того, що точні алгоритми не має сенсу використовувати для задачі складання розкладу.

3.9 Аналіз точності роботи запропонованого для автоматизованого складання розкладу алгоритма

3.9.1 Точність роботи запропонованого для автоматизованого складання розкладу алгоритма

Для оцінки точності було використано верхню оцінку(принцип оцінки рішення-кандидата було описано в розділі 2). Верхня оцінка — це максимальна оцінка, яку може отримати рішення-кандидат, за умови виконання всіх, накладених на нього, обмежень, як обов’язкових ,так і бажаних.

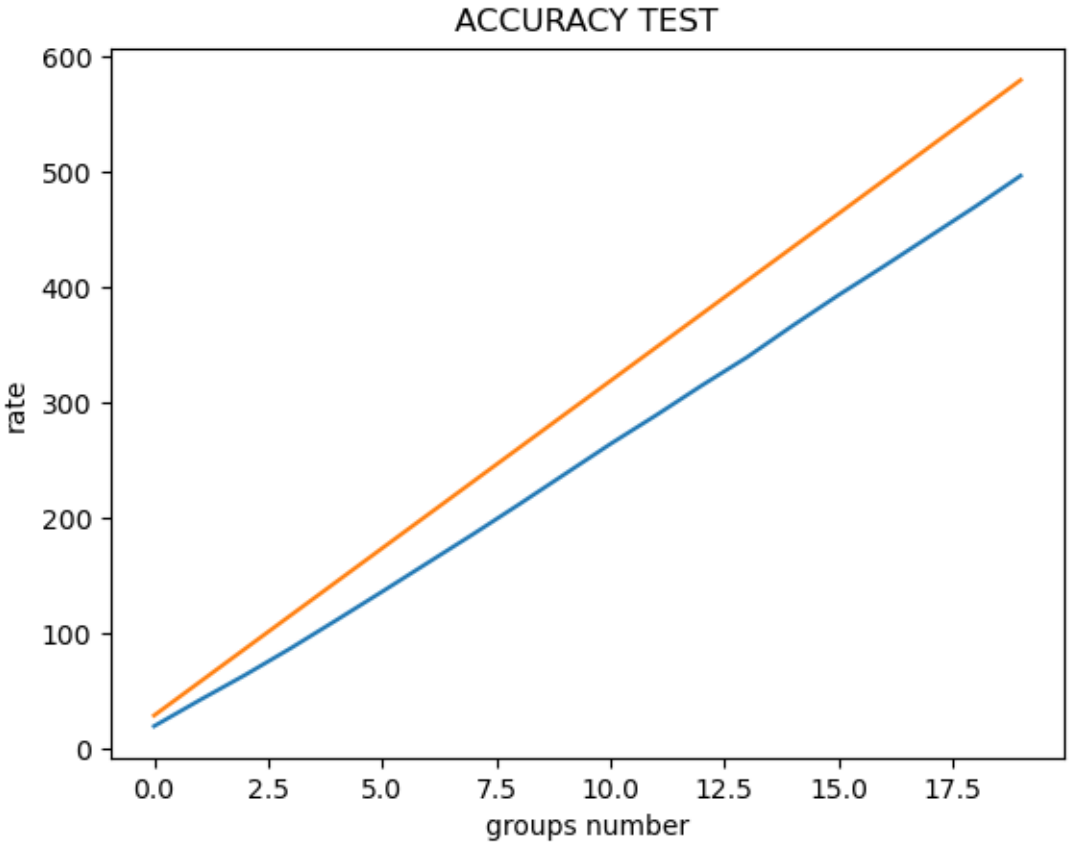

Рисунок 3.28 – Графік точності знайдених рішень для ACS

Проте, на практиці, ймовірність отримати максимальну оцінку для складеного рішення-кандидата прямує до нуля, адже майже неможливо отримати вибірку без протирічь, наприклад, в побажаннях викладачів. На рисунку 3.28 зображено графік точності роботи алгоритма ACS для вибірки, що була описана в пункті 3.8.1. Червоним кольором позначено графік верхньої оцінки, а блакитним — оцінки рішення, що була отримана на практиці.

Опираючись на графіки точності можемо зробити висновок, що алгоритм ACS працює з досить високою точністю.

3.9.2 Порівняння точності роботи запропонованого алгоритма з генетичним алгоритмом

Аналогічно до пункту 3.8, буде навдено тест ефективності генетиного алгоритму.На рисунку нижче зображено графік точності роботи генетичного

алгоритму для вибірки, що була описана в пункті 3.8.2.. Червоним кольором позначено графік верхньої оцінки, а блакитним — оцінки рішення, що була отримана на практиці.

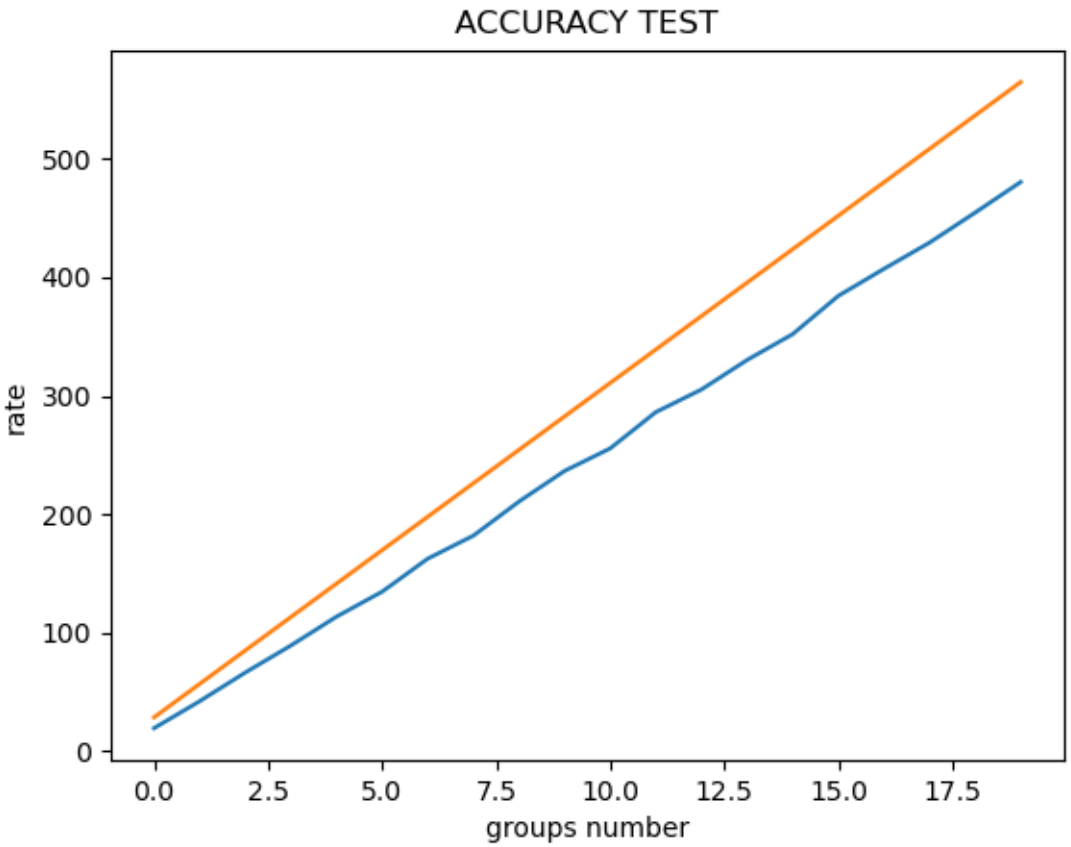

Рисунок 3.29 – Графік точності знайдених рішень для GA

Можемо зробити висновок, що генетичний алгоритм, для задачі складання розкладу, поступається запропонованому алгоритму як за ефективністю, так і за точністю.

3.10 Висновки до розділу 3

Програмний продукт, який було реалізовано в ході дипломної роботи, є зручним інструментом для автоматизованої побудови розкладу. Його функціонал дозволяє внести всю необхідну інформацію про навчальний заклад та на виході отримати інтерактивні таблиці з розкладом для навчальних груп та кожного з викладачів.

Запропонований в дипломній роботі алгоритм є досить точним та ефективним, у порівнянні з генетичним алгоритмом та алгоритмом повного перебору.

4 ФУНКЦІОНАЛЬНО-ВАРТІСНИЙ АНАЛІЗ ПРОГРАМНОГО ПРОДУКТУ

4.1 Постановка задачі проектування

Спроектований програмний продукт для автоматизованого складання розкладу в навчальних закладах. Реалізовано за допомогою мови C++ та бібліотеки Qt.

4.2 Обґрунтування функцій та параметрів програмного продукту

F1 - Внесення інформації про навчальний заклад:

а) завантаження статичних даних

б) введення даних через інтерфейс програми.

F2 - Побудова розкладу:

а) з врахуванням побажань викладачів про час проведення пари

б) без врахування побажань викладачів про час проведення пари

F3 - Побудова алгоритму:

а) з використанням threadPool

б) без використання threadPool

F4 - Отримання створеного розкладу:

а) у вигляді інтерактивної таблиці

б) у вигляді excel файлу

Для характеристики прототипу програмного додатку використовуємо параметри X1 – X5. Визначаємо мінімальні, середні отримуванні та максимально допустимі значення. Згідно з морфологічною картою (рисунок 4.1) було побудовано позитивно-негативну таблицю Табл. 2.

На основі морфологічної таблиці та позитивно-негативної матриці можемо зробити висновок,що при розробці програмного продукту деякі варіанти реалізації функцій варто відкинути, адже вони не відповідають поставленим перед програмним продуктом задачам.

Розглянемо більш деталально кожну з функцій.

Табл. 2 – Позитивно-негативна матриця

Функції	Варіанти реалізації	Переваги	Недоліки
F1	А	Не вимагає втручання людини	Використовується лише для тестування
	Б	Використовується для довільних даних	Вимагає втручання людини
F2	А	Результуючий результат є більш точним	Вимагає додаткового опитування викладачів
	Б	Не вимагає додаткового опитування викладачів	Результуючий результат є менш точним
F3	А	Є більш ефективним по часу	Вимагає додаткового часу на розробку
	Б	Є швидшим в реалізації	Є менш ефективним по часу виконання
F4	А	Дає можливість редагування кінцевого результата	Зберігається локально в додатку
	Б	Зберігання в зручному, для подальшої роботи, форматі	Дає можливість редагування кінцевого результата в додатку

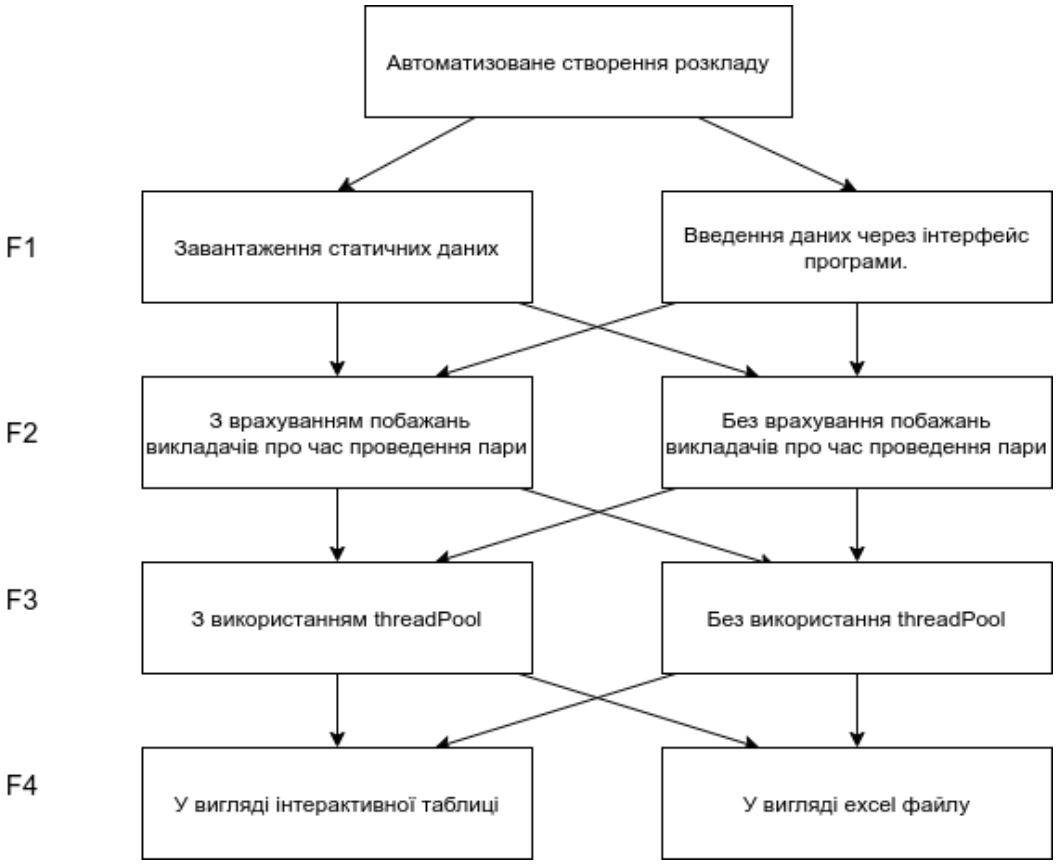

Рисунок 4.1 – Морфологічна карта

1. Функція F_1: Перевагу віддамо введенню даних через інтерфейс користувача, адже такий метод є більш універсальним.
2. Функція F_2: Перевагу віддамо побудові розкладу з урахування побажань викладачів про час проведення пари. З урахуванням цих данних мона отримати більш точний результат.
3. Функція F_3: Перевагу віддамо використанню threadPool. Такий підхід є більш оптимізованим з точки зору часу роботи спроектованого алгоритму.
4. Функція F_4: Обидва варіанти можна використовувати в розробці.

Таким чином, будемо розглядати такі варіанти релазції програмного продукту:

$$F_1b - F_2a - F_3a - F_4a$$

$$F_1b - F_2a - F_3a - F_3b$$

Назва параметра	Позначення	Одиниці виміру	Значення парамтера		
			Гірші	Середні	Кращі
Зручність введення даних	X1	кількість кліків	40	30	25
Точність складеного розкладу	X2	відсотки	50	70	90
Шивдкодія програмного продукту	X3	секунди	10	8	7
Об’єм пам’яті для збереження розкладу	X4	мб	40	30	25

Для оцінювання якості розглянутих функцій обрана система параметрів, описана нижче.

4.3 Обгрунтування системи парамтерів програмного продукту

На основі даних, що були розглянуті вище,визначиио основні параметри вибору, що будуть використані для розрахунку коефіцієнта технічного рівня.

Для того, щоб охарактеризувати програмний продукт, будемо використовувати наступні параметри:

- X_1 — зручність введення даних
- X_2 — точність складеного розкладу
- X_3 — швидкодія програмного продукту
- X_4 — об’єм пам’яті для збереження розкладу

Оцінка значень параметрів наведена в таблиці нижче.

За данними наведеної вище таблиці побудуємо графічні характеристики параметрів $X_1 - X_4$.

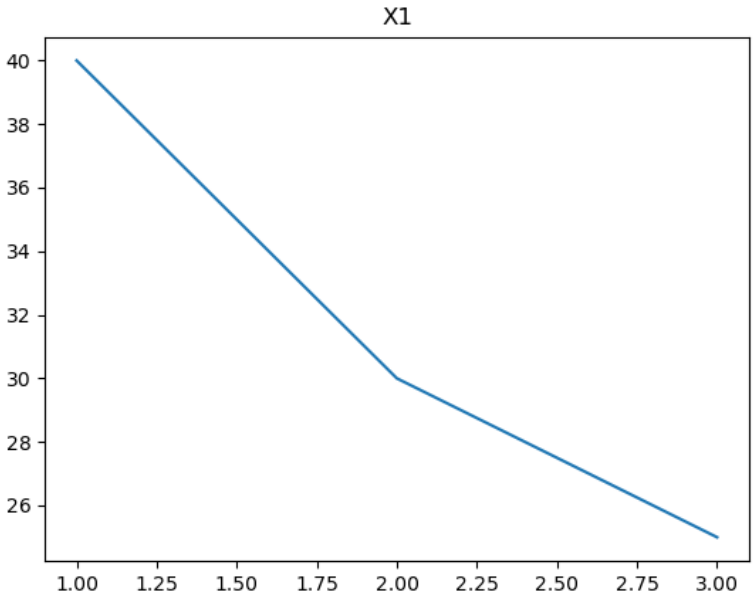

Рисунок 4.2 – X_1, зручність введення даних

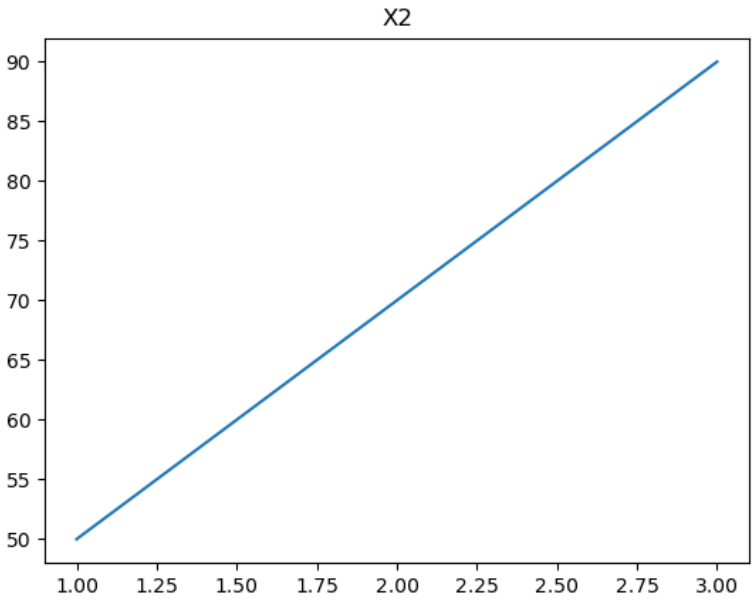

Рисунок 4.3 – X_2, точність складеного розкладу

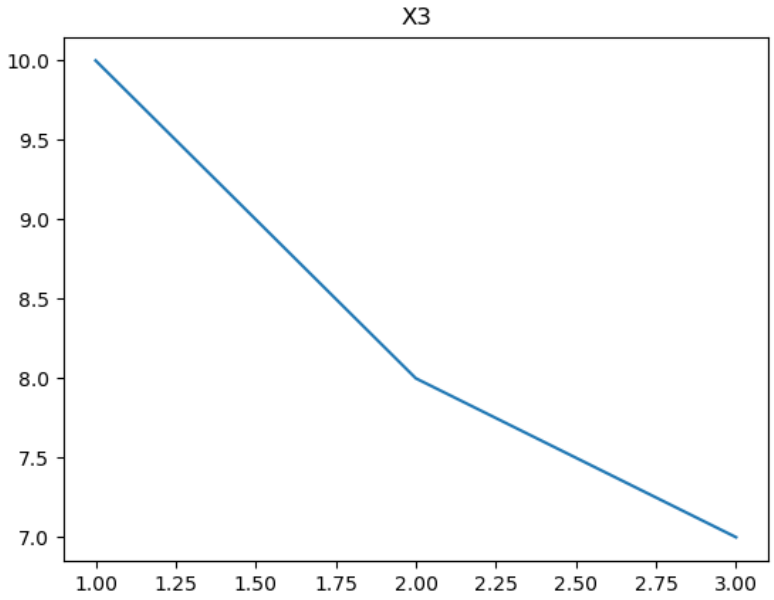

Рисунок 4.4 – X_3, швидкодія програмного продукту

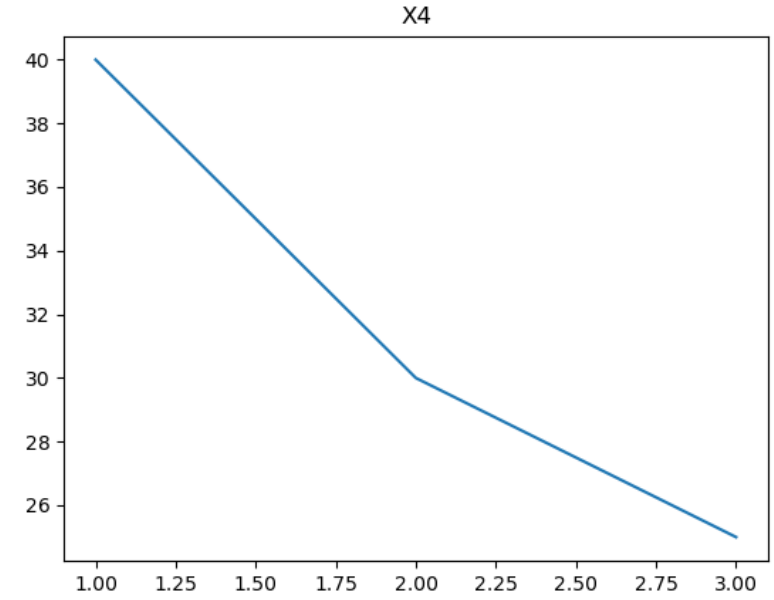

Рисунок 4.5 – X_4, об’єм пам’яті для збереження розкладу

4.4 Аналіз експертного оцінювання параметрів

Методом експертної оцінки, визначимо важливість кожного параметру для розробки програмного продукту для автоматизованого складання розкладу.

Табл. 3 – Результати ранжування параметрів

Назва параметра	Позначення	Одиниці виміру	Ранг параметра за оцінкою експерта							Сума рангів R_i	Відхилення Δ_i	Δ_i^2
			1	2	3	4	5	6	7			
Зручність введення даних	X1	кількість кліків	4	3	4	3	4	3	4	25	7.5	56.25
Точність складеного розкладу	X2	відсотки	1	1	1	1	1	1	2	7	-10.5	110.25
Шивдкодія програмного продукту	X3	секунди	2	2	2	2	3	2	1	15	-2.5	6.25
Об’єм пам’яті для збереження розкладу	X4	мб	3	4	3	4	3	4	3	23	5.5	30.25
Разом			10	10	10	10	10	10	10	10	0	203

Значимість кожного з параметрів визначається методом попарного порівняння. ранги варіюються від 1 до 4, де 1 — найменший, а 4 — найбільший. Результати експертного ранжування наведено в Табл. 3 нижче.

Розрахуємо коєфіцієнт узгодженості:

$$W = \frac{12S}{N^2(n^3 - n)} = \frac{12 * 203}{7^2(4^3 - 4)} > W_k = 0.83$$

Коефіцієнт узгодженості більше нормативного, тому результати можна вважати достовірними.

Скориставшись результатами ранжирування, проведемо попарне порівняння всіх параметрів і результати занесемо у Табл. 4.

Табл. 4 – Попарне зрівняння параметрів

Параметри	Експерти							Кінцева оцінка	Числове значення
	1	2	3	4	5	6	7		
X1 та X2	>	>	>	>	>	>	>	>	1,5
X1 та X3	>	>	>	>	>	>	>	>	1,5
X1 та X4	>	<	>	<	>	<	>	>	1,5
X2 та X3	<	<	<	<	>	<	>	<	0,5
X2 та X4	<	<	<	<	>	<	>	<	0,5
X3 та X4	<	<	<	<	>	<	>	<	0,5

Проведемо розрахунок вагомості параметрів. Результати наведено в Табл. 5 нижче.

Табл. 5 – Розрахунок вагомості параметрів

Пара-метри x_i	Параметри x_j				Перша ітер		Друга ітер		Третя ітер	
	X1	X2	X3	X4	b_i	$K_{\text{ві}}$	b_i	$K_{\text{ві}}$	b_i	$K_{\text{ві}}$
X1	1,0	1,5	1,5	1,5	5,500	0,344	21,250	0,317	89,875	0,324
X2	1,5	1,0	0,5	0,5	3,500	0,219	15,250	0,228	62,375	0,225
X3	1,5	0,5	1,0	0,5	3,500	0,219	15,250	0,228	62,375	0,225
X4	1,5	0,5	0,5	1,0	3,500	0,219	15,250	0,228	62,375	0,225
Всього					16,000	1,000	67,000	1,000	277,000	1,000

4.5 Аналіз рівня якості варіантів реалізації функцій

Визначимо рівень якості кожного варіанта виконання основних функцій окремо. Розрахунки наведено в таблиці нижче.

За даними з таблиці 4 визначимо якість кожного з варіантів.

$$K_{\text{к1}} = 1,93 + 0,68 + 1,96 + 1,80 = 6,37$$
$$K_{\text{к2}} = 1,93 + 0,68 + 1,96 + 0,90 = 5,47 \qquad (4.1)$$

Як видно з розрахунків, кращим є перший варіант, для якого коефіцієнт технічного рівня має найбільше значення.

Табл. 6 – Розрахунок показників рівня якості варіантів реалізації основних функцій програмного продукта

Основна функція	Варіант реалізації	Пара-метри	Абсолютне значення параметра	Бальна оцінка параметра	Коефіцієнт вагомості параметра	Коефі-цієнт якості
F1	Б	X1	153	5,96	0,324	1,93
F2	А	X2	147	3	0,225	0,68
F3	А	X3	130	8,67	0,225	1,96
F4	А	X4	1	8	0,225	1,80
	Б	X4	3	4	0,225	0,90

4.6 Економічний аналіз варіантів розробки програмного продукту

Для визначення вартості розробки програмного продукту спочатку проведемо розрахунок трудомісткості. Всі варіанти включають в себе два окремих завдання:

1. Розробка алгоритму програмного продукту
2. Розробка графічного інтерфейса користувача програмного продукту

Завдання 1 за ступенем новизни відноситься до групи А, завдання 2 – до групи Б. За складністю алгоритми, які використовуються в завданні 1 належать до групи 1; а в завданні 2 – до групи 3. Для реалізації завдання 1 використовується довідкова інформація, а завдання 2 використовує інформацію у вигляді даних. Проведемо розрахунок норм часу на розробку та програмування для кожного з завдань.

$$T_1 = 90 * 1.7 * 0.8 = 122.4 \text{ людино-днів}$$

$$T_2 = 27 * 0.9 * 0.8 = 19.44 \text{ людино-днів}$$

Розрахуємо трудомісткість для кожного з варіантів розробки програми.

$$T_I = (19.44 + 122.4 + 122.4 + 122.4) * 8 = 3093.12 \text{ людино-днів}$$

$$T_{II} = (19.44 + 122.4 + 122.4 + 19.44) * 8 = 2269.44 \text{ людино-днів}$$

Найбільшу трудомісткість має варіант 1.

Нехай в розробці беруть участь 3 програмісти з окладом в 15’000 грн. Визначимо середню заробітну плату за годину.

$$СЧ = \frac{3 * 15'000}{3 * 21 * 8} = 89.29 \text{ грн}$$

Заробітна плата для кожного з варіантів реалізації:

$$C_{1зп} = 89.29 * 3093.12 = 276'184.69 \text{ грн} \quad (4.2)$$

$$C_{2зп} = 89.29 * 2269.44 = 202'638.3 \text{ грн} \quad (4.3)$$

Відрахування на соціальне страхування(22%):

$$C_{1від} = 276'184.685 * 0,22 = 60'759.97 \text{ грн} \quad (4.4)$$

$$C_{2від} = 202'638.297 * 0,22 = 44'580.43 \text{ грн} \quad (4.5)$$

Визначимо витрати на оплату однієї машино-години. 3 ЕОМ обслуговують 3 програмістів з окладом 15’000 грн та коефіцієнтом зайнятості 0.47, для однієї ЕОМ отримаємо.

$$C_Г = 12 * 15000 * 0,47 = 84‘600 \text{ грн} \quad (4.6)$$

Враховуючи додаткову заробітню платню (25%)

$$C_{зп} = 84'600 * (1 + 0,25) = 105'750 \text{ грн} \quad (4.7)$$

Відрахування на соціальне страхування(22%)

$$C_{від} = 105'750 * 0,22 = 23'265 \text{ грн} \quad (4.8)$$

Розрахуємо амортизаційні підрахунки (амортизація 25%, вартість ЕОМ 30000 грн)

$$C_а = К_{тм} * К_а * Ц_{пр} = 1,15 * 0,25 * 30000 = 8'625 \text{ грн} \quad (4.9)$$

Розрахуємо витрати на ремонт та профілактику

$$C_р = K_{тм} * K_р * Ц_{пр} = 1,15 * 0,05 * 30000 = 1725 \text{ грн} \quad (4.10)$$

Розрахуємо ефективний годинний фонд часу ПК за рік

$$T_{еф} = (365 - 138 - 16) * 0,8 * 8 = 1350,4 \text{ год} \quad (4.11)$$

Розрахуємо витрати на електроенергію

$$C_{ел} = 1350.4 * 0.3 * 0.7 * 3.51 = 964,95 \text{ грн} \quad (4.12)$$

Накладні витрати дорівнюють:

$$C_н = 30000 * 0,67 = 20'100 \text{ грн} \quad (4.13)$$

Отже експлуатаційні витрати:

$$C_{екс} = 105'750 + 23'625 + 8'625 + 1725 + 964,95 + 20'100 = 160'789.95 \text{ грн} \quad (4.14)$$

Тоді собівартість однієї машино-години ЕОМ дорівнюватиме:

$$C_{м\text{-}г} = \frac{160'789.95}{1350.4} = 119,06 \quad (4.15)$$

Враховуючи, що всі роботи ведуться на ЕОМ, витрати на оплату машинного часу:

$$C_{м1} = 119,06 * 3093.12 = 370'679.5 \text{ грн} \quad (4.16)$$

$$C_{м2} = 119,06 * 2269.44 = 271'969.69 \text{ грн} \quad (4.17)$$

Накладні витрати складають 67% від заробітної плати:

$$C_{н1} = 276'184.69 * 0,67 = 185'043.74 \text{ грн} \quad (4.18)$$

$$C_{н2} = 202'638.3 * 0,67 = 182'219.692 \text{ грн} \quad (4.19)$$

Отже, вартість розробки ПП за варіантами становить:

$$C_{пп1} = 276'184.69 + 60'759.97 + 370'679.5 + 185'043.74 = 892'667.9 \text{ грн} \quad (4.20)$$

$$C_{пп2} = 202'638.3 + 44'580.43 + 271'969.69 + 182'219.692 = 701'408.112 \text{ грн} \quad (4.21)$$

4.7 Вибір кращого варіанта програмного продукту техніко-економічного рівня

$$K_{\text{к1}} = 6,37$$

$$K_{\text{к2}} = 5,47$$

$$K_{mep1} = \frac{6,37}{892'667.9} = 7.136 * 10^{-6} \tag{4.22}$$

$$K_{mep2} = \frac{5,47}{701'408.112} = 7,79 * 10^{-6} \tag{4.23}$$

4.8 Висновки до розділу 4

Виходячи з проведених розрахунків, можемо зробити висновок, що найбільш ефективним буде варіант розробки 2 з коефіцієнтом техніко-економічного рівня $7,79 * 10^{-6}$. Таким чином, після проведеного функціонально – вартісного аналізу було прийняте рішення реалізовувати варіант з введенням даних через користувацький інтерфейс під час роботи програми, побудова розкладу з урахуванням побажань викладачів про час проведення заняття, використиання ідіоми threadPool та збереження отриманого результату у зручному для користувача форматі.

ВИСНОВКИ

В даній дипломній роботі було розглянуто проблему атоматизованого складання розкладу для навчальних закладів.

Задача складання розкладу є NP-складною, тому точні алгоритми використовувати не має практичного сенсу. Провівши порівняльний аналіз евристичних алгоритмів, було прийнято рішення в основу алгоритма для вирішення поставленої задачі покласти алгоритм систем мурашиних колоній(ACS). ACS — евристичний алгоритм ройового інтелекту для пошуку найбільш оптимального шляху в графі. Модифікований алгоритм працює в паралельному середовищі та показує хороші результати в точності та швидкості роботи у порівнянні з найбільш поширеним, для задач подібного типу, генетичним алгоритмом.

В ході роботи було проведено порівняльний аналіз комерційних програмних продуктів, що представлені на ринку на даний момент, викоремлено їх переваги та недоліки. На основі цього аналізу було сформовано вимоги до програмного продукту. Було створено додаток для автоматизованого складання розкладу. Його функціонал дозволяє внести інформацію про навчальний заклад,а саме про викладачів, навчальні предмети, навчальні групи та аудиторний фонд, а на виході отримати інтерактивну таблицю з розкладом для навчальних груп та для кожного з викладачів.

ЛІТЕРАТУРА

1. Cormen T., Leiserson C., Rivest R., Stein C. Introduction to Algorithms The MIT Press, 2009. 1313 c.

2. Docendo [Електронний ресурс] — Режим доступу до ресурсу: https://docendo.co/ 20.04.2021

3. QuickSchool [Електронний ресурс] — Режим доступу до ресурсу: https://www.quickschools.com/ 20.04.2021

4. aSc timetables [Електронний ресурс] — Режим доступу до ресурсу: https://www.asctimetables.com/ 20.04.2021

5. A review on genetic algorithm: past, present, and future. Sourabh Katoch, Sumit Singh Chauhan, Vijay Kumar Published: 31 October 2020

6. Evolutionary Programming and Evolution Strategies: Similarities and Differences · Thomas Bck, G. Rudolph, H. Schwefel · Published 1993

7. Docendo [Електронний ресурс] — Режим доступу до ресурсу: https://www.alanzucconi.com/2016/04/06/evolutionary-coputation-1 28.04.2021

8. Enabled Energy Management for Electrified Vehicles, Clara Marina Martínez, Dongpu Cao, in Ihorizon 2019

9. Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem. M. Dorigo, L. M. Gambardella, 1997.

10. Ant System: Optimization by a Colony of Cooperating Agents. M. Dorigo, V. Maniezzo, A. Colorni, 1996.

11. Шлее М. Qt 5.10 Профессиональное программирование на C++. Санкт-Петербург: БХВПетербург, 2018. 1072 с.

ДОДАТОК

ДОДАТОК А
ЛІСТИНГ КОДУ

```
#ifndef ANT_H
#define ANT_H
#include <variant>

#include ”AntColony.h”
#include ”UniversityData.h”

using SolutionCandidateType =
    std::vector<std::pair<std::shared_ptr<Vertex>, std::shared_ptr<
       Class>>>;
using VertexType = std::vector<std::pair<std::shared_ptr<Vertex>,
   uint>>;

// structure for second heuristic
struct Vertex_ {
  std::shared_ptr<Vertex> v;
  uint index;
  double rate = 1;

  Vertex_() {}
  Vertex_(std::shared_ptr<Vertex> vert, uint idx) : v(vert), index(
     idx) {}
};

class AntColony;
class Ant {
  std::weak_ptr<AntColony> base;
  uint antIndex;
  uint groupIndex;

  SolutionCandidateType solution;
  std::vector<uint> solutionIndexSeq;
  bool validSolution = true;
```

```
31
32    std::shared_ptr<Vertex> currentVertex = nullptr;
33    std::shared_ptr<Class> currentClass = nullptr;
34    uint currentIndex = 0;
35    uint gapNum = 0;
36    uint gapIndex;
37
38    void getClass();
39
40    // first heuristic
41    void getNextVertex_h1();
42    VertexType remainVertexes;
43
44    // second heuristic
45    std::vector<Vertex_> remainVert_;
46    void getNextVertex_h2();
47
48    // on-fly rate methods
49    double rateTeacherPref(uint idx);
50    double rateStudentsGap(uint idx);
51    double rateLPCombination(uint idx);
52    double ratelectureOrder(uint idx);
53    void rateNext(uint idx);
54
55  public:
56    Ant() {}
57    Ant(std::shared_ptr<AntColony> b, uint antId, uint groupId);
58
59    void createSolutionCandidate();
60
61    std::vector<uint> getSolutionIndexSeq() const;
62    std::variant<SolutionCandidateType, bool> getSolution() const;
63
64    const VertexType &getRemainVertexes() const;
65
66    ~Ant();
67
```

```
68    friend class AntColony;
69  };
70
71  #endif  // ANT_H
```

```
 1  #include ”Ant.h”
 2
 3  #include <cassert>
 4  #include <random>
 5
 6  #include ”AntSystem.h”
 7
 8  #define H2
 9
10  barrier general_barrier;
11  barrier vertex_barrier;
12
13  template <typename Numeric, typename Generator = std::mt19937>
14  Numeric random(Numeric from, Numeric to) {
15    thread_local static Generator gen(std::random_device{}());
16
17    using dist_type =
18        typename std::conditional<std::is_integral<Numeric>::value,
19                                  std::uniform_int_distribution<Numeric
                                      >,
20                                  std::uniform_real_distribution<
                                      Numeric>>::type;
21
22    thread_local static dist_type dist;
23
24    return dist(gen, typename dist_type::param_type{from, to});
25  }
26
27  Ant::Ant(std::shared_ptr<AntColony> b, uint antId, uint groupId)
28      : base(b), antIndex(antId), groupIndex(groupId) {
29    auto size = base.lock()->ud.groups[groupIndex]->getSubjectGraphSize
          ();
30    gapIndex = size + 1;
```

```
31
32 #ifdef H1
33   remainVertexes.resize(size + 1);
34   for (uint i = 0; i < size; ++i) {
35     remainVertexes[i] = std::make_pair(
36         base.lock()->ud.groups[groupId]->getSubjectGraph()->getVertex
           (i),
37         i + 1);
38   }
39
40   remainVertexes[size] = {std::shared_ptr<Vertex>(new Vertex),
      gapIndex};
41
42 #endif
43
44 #ifdef H2
45   remainVert_.resize(size + 1);
46   for (uint i = 0; i < size; ++i) {
47     remainVert_[i] = Vertex_(
48         base.lock()->ud.groups[groupId]->getSubjectGraph()->getVertex
           (i),
49         i + 1);
50   }
51
52   remainVert_[size] = Vertex_(std::shared_ptr<Vertex>(new Vertex),
      gapIndex);
53
54 #endif
55   gapNum = workDays * workHours -
56           base.lock()->ud.groups[groupIndex]->getSubjectGraphSize();
57
58   solutionIndexSeq.push_back(currentIndex);
59 }
60
61 std::vector<uint> Ant::getSolutionIndexSeq() const { return
     solutionIndexSeq; }
62
```

```
63 std::variant<SolutionCandidateType, bool> Ant::getSolution() const {
64   std::variant<SolutionCandidateType, bool> s;
65   if (validSolution) {
66     s = solution;
67   } else {
68     s = false;
69   }
70   return s;
71 }
72
73 const VertexType& Ant::getRemainVertexes() const { return
      remainVertexes; }
74
75 void Ant::createSolutionCandidate() {
76 #ifdef H1
77   while (!remainVertexes.empty() && validSolution) {
78     getNextVertex_h1();
79 #endif
80
81 #ifdef H2
82     while (!remainVert_.empty() && validSolution) {
83       getNextVertex_h2();
84 #endif
85       if (validSolution) {
86         if (currentIndex != gapIndex) {
87           getClass();
88           solution.push_back({currentVertex, currentClass});
89         } else {
90           solution.push_back({currentVertex, nullptr});
91         }
92         solutionIndexSeq.push_back(currentIndex);
93       }
94
95       general_barrier.wait();
96
97       if (validSolution && currentIndex != gapIndex) {
98         currentVertex->teacher->release(antIndex);
```

```
99          currentClass->release(antIndex);
100         currentClass = nullptr;
101       }
102     }
103     vertex_barrier.wait(true);
104     general_barrier.wait(true);
105   }
106
107   void Ant::getClass() {
108     auto base_shared = base.lock();
109
110     auto getClass = [this, base_shared](uint antIndex, uint classType
          ) {
111       auto it = base_shared->ud.classes[classType].begin();
112       while (it != base_shared->ud.classes[classType].end()) {
113         if ((*it)->try_get(antIndex)) {
114           currentClass = *it;
115           return;
116         }
117         ++it;
118       }
119     };
120
121     if (currentVertex->subject->getLabClass()) {
122       getClass(antIndex, 2);
123     } else if (!currentVertex->subject->getType()) {
124       getClass(antIndex, 0);
125
126     } else if (currentVertex->subject->getType() ||
127                (base.lock()->ud.groups[groupIndex]->getCapacity() <=
128                     labClassCapacity &&
129                 currentClass == nullptr)) {
130       getClass(antIndex, 1);
131
132       if (currentClass == nullptr) {
133         getClass(antIndex, 0);
134       }
```

```
135     }
136
137     if (currentClass == nullptr) {
138       validSolution = false;
139     }
140   }
141
142   void Ant::getNextVertex_h1() {
143     VertexType vertexBackup;
144     auto base_shared = base.lock();
145
146     double q = random<double>(0.0, 1.);
147     if (q <= q0) {
148       std::sort(
149           remainVertexes.begin(), remainVertexes.end(),
150           [this, base_shared](const auto& lhs, const auto& rhs) {
151             return (
152                 pow(base_shared->pheromone[currentIndex][lhs.second],
                        alpha) >
153                 pow(base_shared->pheromone[currentIndex][rhs.second],
                        alpha));
154           });
155
156     } else {
157       auto sum = std::accumulate(
158           remainVertexes.begin(), remainVertexes.end(), 0.0,
159           [this, base_shared](auto s, const auto& cur) {
160             return s +
161                    pow(base_shared->pheromone[currentIndex][cur.
                          second], alpha);
162           });
163
164       std::sort(remainVertexes.begin(), remainVertexes.end(),
165                 [this, sum, base_shared](const auto& lhs, const auto&
                        rhs) {
166                   return pow(base_shared->pheromone[currentIndex][lhs
                          .second],
```

```
167                                 alpha) /
168                                 sum >
169                             pow(base_shared->pheromone[currentIndex][rhs
                                    .second],
170                                 alpha) /
171                                 sum;
172                 });
173     }
174
175     vertex_barrier.wait();
176     bool gotVertex = false;
177     while (!remainVertexes.empty() && !gotVertex) {
178       if (remainVertexes.begin()->second == gapIndex) {
179         base_shared->updatePheromone(currentIndex, gapIndex, ksi);
180
181         currentVertex = remainVertexes.begin()->first;
182         currentIndex = remainVertexes.begin()->second;
183
184         --gapNum;
185         gotVertex = true;
186         if (gapNum == 0) {
187           remainVertexes.erase(remainVertexes.begin());
188         }
189       } else if (remainVertexes.begin()->first->teacher->try_get(
              antIndex)) {
190         base_shared->updatePheromone(currentIndex,
191                                      remainVertexes.begin()->second,
                                        ksi);
192
193         currentVertex = remainVertexes.begin()->first;
194         currentIndex = remainVertexes.begin()->second;
195
196         gotVertex = true;
197
198         remainVertexes.erase(remainVertexes.begin());
199       } else {
200         vertexBackup.push_back(*remainVertexes.begin());
```

```
201         remainVertexes.erase(remainVertexes.begin());
202       }
203     }
204
205     if (!gotVertex) {
206       validSolution = false;
207       return;
208     }
209
210     for (auto& b : vertexBackup) {
211       remainVertexes.push_back(b);
212     }
213   }
214
215   void Ant::getNextVertex_h2() {
216     double q = random<double>(0.0, 1.);
217     std::vector<Vertex_> vertexBackup;
218     auto base_shared = base.lock();
219     for (uint i = 0; i < remainVert_.size(); ++i) {
220       rateNext(i);
221     }
222     if (q <= q0) {
223       std::sort(
224           remainVert_.begin(), remainVert_.end(),
225           [this](const auto& lhs, const auto& rhs) {
226             auto base_shared = base.lock();
227             return (
228                 pow(base_shared->pheromone[currentIndex][lhs.index],
                       alpha) *
229                     pow((-1.) / lhs.rate, beta) >
230                 pow(base_shared->pheromone[currentIndex][rhs.index],
                       alpha) *
231                     pow((-1.) / rhs.rate, beta));
232           });
233
234     } else {
235       auto sum = std::accumulate(
```

```
236           remainVert_.begin(), remainVert_.end(), 0.0,
237           [this, base_shared](auto s, const auto& cur) {
238             return s +
239                    pow(base_shared->pheromone[currentIndex][cur.index
                       ], alpha) *
240                     pow((1.) / cur.rate, beta);
241           });
242
243       std::sort(
244           remainVert_.begin(), remainVert_.end(),
245           [this, sum](const auto& lhs, const auto& rhs) {
246             auto base_shared = base.lock();
247
248             return (
249                 ((pow(base_shared->pheromone[currentIndex][lhs.index
                      ], alpha) *
250                   pow((-1.) / lhs.rate, beta)) /
251                  sum) >
252                 ((pow(base_shared->pheromone[currentIndex][rhs.index
                      ], alpha) *
253                   pow((-1.) / rhs.rate, beta)) /
254                  sum));
255           });
256     }
257
258     vertex_barrier.wait();
259
260     bool gotVertex = false;
261     while (!remainVert_.empty() && !gotVertex) {
262       if (remainVert_.begin()->index == gapIndex) {
263         base_shared->updatePheromone(currentIndex, gapIndex,
264                                      remainVert_.begin()->rate);
265
266         currentVertex = remainVert_.begin()->v;
267         currentIndex = remainVert_.begin()->index;
268
269         --gapNum;
```

```
270         gotVertex = true;
271
272         if (gapNum == 0) {
273           remainVert_.erase(remainVert_.begin());
274         }
275
276       } else if (remainVert_.begin()->v->teacher->try_get(antIndex))
             {
277         base_shared->updatePheromone(currentIndex, remainVert_.begin
              ()->index,
278                                      remainVert_.begin()->rate);
279
280         currentVertex = remainVert_.begin()->v;
281         currentIndex = remainVert_.begin()->index;
282
283         gotVertex = true;
284
285         remainVert_.erase(remainVert_.begin());
286       } else {
287         vertexBackup.push_back(*remainVert_.begin());
288         remainVert_.erase(remainVert_.begin());
289       }
290     }
291
292     if (!gotVertex) {
293       validSolution = false;
294       return;
295     }
296
297     for (auto& b : vertexBackup) {
298       remainVert_.push_back(b);
299     }
300   }
301
302   double Ant::rateTeacherPref(uint idx) {
303     double rate = 0.;
304     if (remainVert_[idx].index == gapIndex) {
```

```
305       rate += gap_rate;
306     } else if (remainVert_[idx].v->teacher->getPref(solution.size()))
            {
307       rate += 1;
308     }
309
310     return rate * teacher_pref_coef;
311   }
312
313   double Ant::rateStudentsGap(uint idx) {
314     double rate = 0;
315
316     if (remainVert_[idx].index == gapIndex &&
317         (solution.size() % workHours == 0 ||
318          solution.size() % workHours == 4)) {
319       rate += 1;
320     }
321
322     return rate * students_gap_coef;
323   }
324
325   double Ant::rateLPCombination(uint idx) {
326     double rate = 0;
327
328     if (currentVertex == nullptr || remainVert_[idx].index ==
           gapIndex ||
329         currentIndex == gapIndex) {
330       return 0;
331     }
332     if (std::find_if(
333             solution.begin() + ((solution.size() / workDays) *
                workHours),
334             solution.end(), [this, idx](const auto& current) {
335               return current.first->subject == remainVert_[idx].v->
                     subject;
336             }) == solution.end()) {
337       rate += 1;
```

```
338     }
339
340     if (currentVertex->subject->getTitle() ==
341             remainVert_[idx].v->subject->getTitle() &&
342         currentVertex->subject->getType() !=
343             remainVert_[idx].v->subject->getType()) {
344       rate += 1;
345     }
346
347     return rate * lp_combination_coef;
348   }
349
350   double Ant::ratelectureOrder(uint idx) {
351     if (currentVertex == nullptr || remainVert_[idx].index ==
          gapIndex ||
352         currentIndex == gapIndex) {
353       return 0;
354     }
355
356     double rate = 0.;
357     uint pairNum = solution.size() % workHours;
358     if ((pairNum >= 2) && !remainVert_[idx].v->subject->getType()) {
359       rate += 1;
360     } else if (pairNum <= 1 && remainVert_[idx].v->subject->getType()
          ) {
361       rate += 1;
362     }
363
364     return rate * lecture_order_coef;
365   }
366
367   void Ant::rateNext(uint idx) {
368     double currentRate = 0;
369     // no same subjects in one day, lecture and practice in the same
          day
370     currentRate += rateLPCombination(idx);
371
```

```
372     // teacher prefs and little gap rate
373     currentRate += rateTeacherPref(idx);
374
375     //+1 if gap is first pair or last
376     currentRate += rateStudentsGap(idx);
377
378     // +1 lecture 1,2,3
379     // +1 practice 1,4,5
380     currentRate += ratelectureOrder(idx);
381     remainVert_[idx].rate = currentRate;
382   }
383
384   Ant::~Ant() {}
```

```
 1 #ifndef ANTCOLONY_H
 2 #define ANTCOLONY_H
 3 #include <memory>
 4 #include <thread>
 5
 6 #include ”Ant.h”
 7 #include ”Common.h”
 8 #include ”UniversityData.h”
 9
10 class Ant;
11 class AntColony : public std::enable_shared_from_this<AntColony> {
12   std::vector<Ant> ants;
13
14   uint colonyIndex;
15   UniversityData &ud;
16
17   matrix pheromone;
18   matrix heuristic;
19
20   void initMatrices();
21
22  public:
23   std::mutex pheromoneUpdateMutex;
24   std::mutex heuristicsUpdateMutex;
```

```
25
26    AntColony(uint colonyId, UniversityData &ud);
27
28    void createAnts();
29
30    void construct(std::vector<std::shared_ptr<std::thread>>::iterator
          first,
31                   std::vector<std::shared_ptr<std::thread>>::iterator
                       last);
32
33    void updatePheromone(uint currentIndex, uint nextIndex, double rate
          );
34    void evaporatePheromone();
35    void globalUpdatePheromone(const std::vector<uint> &indexSeq,
          double rate);
36
37    std::vector<uint> getSolutionIndexSeq(uint antIndex) const;
38
39    ~AntColony();
40
41    friend class Ant;
42    friend class AntSystem;
43 };
44
45 #endif  // ANTCOLONY_H
```

```
1  #include ”AntColony.h”
2
3  AntColony::AntColony(uint colonyId, UniversityData &ud)
4      : colonyIndex(colonyId), ud(ud) {
5    initMatrices();
6  }
7
8  void AntColony::initMatrices() {
9    auto size = workDays * workHours + 1;
10   heuristic.resize(size);
11   pheromone.resize(size);
12
```

```
13    for (auto i = 0u; i < size; ++i) {
14      heuristic[i].resize(size);
15      pheromone[i].resize(size);
16
17      for (auto j = 0u; j < size; ++j) {
18        heuristic[i][j] = 1;
19        pheromone[i][j] = teta0;
20      }
21    }
22  }
23
24  void AntColony::createAnts() {
25    ants.resize(antsNum);
26    for (uint i = 0; i < antsNum; ++i) {
27      ants[i] = Ant(shared_from_this(), i, colonyIndex);
28    }
29  }
30
31  void AntColony::construct(
32      std::vector<std::shared_ptr<std::thread>>::iterator first,
33      std::vector<std::shared_ptr<std::thread>>::iterator last) {
34    createAnts();
35
36    for (auto it = first; it != last; ++it) {
37      *it = std::shared_ptr<std::thread>(
38          new std::thread(&Ant::createSolutionCandidate, &ants[(it -
               first)]));
39    }
40  }
41
42  void AntColony::updatePheromone(uint currentIndex, uint nextIndex,
43                                  double rate) {
44    std::lock_guard lg(pheromoneUpdateMutex);
45    pheromone[currentIndex][nextIndex] += ksi * rate;
46  }
47
48  void AntColony::evaporatePheromone() {
```

```
49    for (auto &r : pheromone) {
50      for (auto &el : r) {
51        el *= (1 - rho);
52      }
53    }
54  }
55
56  void AntColony::globalUpdatePheromone(const std::vector<uint> &
        indexSeq,
57                                        double rate) {
58    if (rate == 0 || indexSeq.empty()) {
59      return;
60    }
61
62    for (uint j = 0; j < indexSeq.size() - 1; ++j) {
63      pheromone[indexSeq[j]][indexSeq[j + 1]] += ksi_g * rate;
64
65      //    pheromone[indexSeq[j]][indexSeq[j + 1]] += ksi_g * rate /
66      //    ud.maxPosibleRate;
67    }
68  }
69
70  std::vector<uint> AntColony::getSolutionIndexSeq(uint antIndex) const
        {
71    return ants[antIndex].getSolutionIndexSeq();
72  }
73
74  AntColony::~AntColony() {}
```

```
1  #ifndef ANTSYSTEM_H
2  #define ANTSYSTEM_H
3  #include <memory>
4  #include <thread>
5  #include <vector>
6
7  #include "AntColony.h"
8  #include "Solution.h"
9  #include "UniversityData.h"
```

```
10 #include ”barrier.h”
11
12 extern barrier general_barrier;
13 extern barrier vertex_barrier;
14
15 class AntSystem {
16   UniversityData& univerData;
17   std::vector<std::shared_ptr<AntColony>> colonies;
18   std::vector<std::shared_ptr<std::thread>> threads;
19
20   std::vector<Solution> solutionCandidates;
21   std::pair<Solution, double> allTimeBest;
22   std::pair<Solution, double> iterBest;
23   std::vector<std::vector<uint>> allTimeBestIndexes;
24
25   const double sigma = 0.1;
26   uint done = 10;
27   uint bestSolutionIterNum;
28
29   void createColonies();
30   void constructTeacherSolution(std::vector<Solution>&
       solutionCandidates);
31   void getStudentsSolution();
32
33   // rate methods
34   double rateCandidate(const Solution& s);
35   double rateTeachersPreferences(const Solution& candidate);
36   double rateLecturesOrder(const Solution& candidate);
37   double rateLPCombinations(const Solution& candidate);
38   double rateStudentsGaps(const Solution& candidate);
39   void getIterBest();
40
41   // verbose info
42
43   uint iterCounter = 0;
44   std::vector<double> plot;
45   void printMatrix();
```

```
46    void dummyTest();
47
48    // brutte force methods
49    void generateSolutions(std::vector<std::vector<uint>> indexes, uint
          index,
50                           std::vector<Solution>& soultions);
51    bool validateSolutions(const Solution& solution);
52    std::shared_ptr<Class> getClass(std::shared_ptr<Vertex>& vert,
53                                    const Solution& s, uint j);
54
55   public:
56    AntSystem(UniversityData& ud);
57    void run();
58    void brutteForce();
59
60    void printStudentsSchadule(const Solution& s);
61    void printTeachersSchadule(const Solution& s);
62
63    Solution getResult() const;
64    uint getIterationsNumber() const;
65    double getSolutionRate() const;
66
67    void dummyPrint();
68
69    ~AntSystem();
70    uint getBestSolutionIterNum() const;
71  };
72
73  #endif  // ANTSYSTEM_H
```

```
1  #include ”AntSystem.h”
2
3  #include <algorithm>
4  #include <cmath>
5  #include <iomanip>
6  #include <numeric>
7
8  AntSystem::AntSystem(UniversityData &ud) : univerData(ud) {}
```

```
 9
10 uint AntSystem::getBestSolutionIterNum() const { return
       bestSolutionIterNum; }
11
12 void AntSystem::createColonies() {
13   colonies.resize(univerData.groups.size());
14   for (uint i = 0; i < univerData.groups.size(); ++i) {
15     colonies[i] = std::shared_ptr<AntColony>(new AntColony(i,
           univerData));
16   }
17 }
18
19 void AntSystem::constructTeacherSolution(
20     std::vector<Solution> &solutionCandidates) {
21   for (auto &c : solutionCandidates) {
22     c.teacherSchadule.resize(univerData.teachers.size());
23     for (auto &t : c.teacherSchadule) {
24       t.resize(workHours * workDays);
25       std::fill(t.begin(), t.end(), TeacherCell(nullptr, nullptr,
             nullptr));
26     }
27   }
28
29   for (auto &candiate : solutionCandidates) {
30     for (auto k = 0u; k < univerData.teachers.size(); ++k) {
31       for (auto i = 0u; i < candiate.studentsSchadule.size(); ++i) {
32         for (auto j = 0u; j < candiate.studentsSchadule[i].size(); ++
               j)
33           if (candiate.studentsSchadule[i][j].teacher ==
34               univerData.teachers[k]) {
35             candiate.teacherSchadule[k][j] = TeacherCell(
36                 candiate.studentsSchadule[i][j].classroom, univerData
                       .groups[i],
37                 candiate.studentsSchadule[i][j].subject);
38           }
39       }
40     }
```

```
41    }
42  }
43
44  void AntSystem::getStudentsSolution() {
45    for (auto i = 0u; i < antsNum; ++i) {
46      Solution s;
47      s.antIndex = i;
48      s.studentsSchadule.resize(univerData.groups.size());
49
50      for (auto &c : s.studentsSchadule) {
51        c.resize(workDays * workHours);
52      }
53
54      bool valid = true;
55      for (auto j = 0u; j < colonies.size() && valid; ++j) {
56        auto sc = colonies[j]->ants[i].getSolution();
57
58        std::visit(
59            [&s, &valid, j](const auto &current) {
60              using T = std::decay_t<decltype(current)>;
61              if constexpr (std::is_same_v<T, bool>) {
62                valid = false;
63                return;
64              } else {
65                s.setStudentsSchadule(current, j);
66              }
67            },
68            sc);
69      }
70
71      if (valid) {
72        solutionCandidates.push_back(s);
73      }
74    }
75  }
76
77  double AntSystem::rateCandidate(const Solution &s) {
```

```
78    double rate = 0;
79    rate += rateTeachersPreferences(s);
80    //  rate += rateLecturesOrder(s);
81    rate += rateStudentsGaps(s);
82    rate += rateLPCombinations(s);
83    return rate;
84  }
85
86  double AntSystem::rateTeachersPreferences(const Solution &candidate)
        {
87    double rate = 0;
88
89    for (auto i = 0u; i < univerData.teachers.size(); ++i) {
90      for (auto j = 0u; j < candidate.teacherSchadule[i].size(); ++j) {
91        if (candidate.teacherSchadule[i][j].group != nullptr &&
92            univerData.teachers[i]->getPreferencesMatrix()[j]) {
93          rate += 1;
94        }
95      }
96    }
97
98    return teacher_pref_coef * rate;
99  }
100
101 double AntSystem::rateLecturesOrder(const Solution &candidate) {
102   double rate = 0;
103   for (const auto &g : candidate.studentsSchadule) {
104     for (const auto &s : univerData.subjects) {
105       auto first = std::find_if(
106           g.begin(), g.end(),
107           [s](const StudentCell &cell) { return cell.subject == s; })
                ;
108       if (first != g.end() && first->subject != nullptr)
109         if (!first->subject->getType()) {
110           rate += 1;
111         }
112     }
```

```
113   }
114   return lecture_order_coef * rate;
115 }
116
117 double AntSystem::rateLPCombinations(const Solution &candidate) {
118   double rate = 0;
119   for (const auto &g : candidate.studentsSchadule) {
120     for (auto it = g.begin(); it != g.end(); it += workHours) {
121       // no same classes in one day for one group
122       uint counter = 0;
123       for (auto i = it; i != it + workHours; ++i) {
124         if (std::find_if(i + 1, it + workHours,
125                          [i, candidate](const auto &current) {
126                            return current.subject == i->subject;
127                          }) != (it + workHours)) {
128           ++counter;
129         }
130
131         // lecture and practice in the same day
132         if (std::find_if(
133                 i + 1, it + workHours, [i, candidate](const auto &
                    current) {
134                   if (current.subject != nullptr && i->subject !=
                      nullptr &&
135                       current.subject->getTitle() == i->subject->
                          getTitle() &&
136                       current.subject->getType() != i->subject->
                          getType())
137                     return true;
138                   else
139                     return false;
140                 }) != (it + workHours)) {
141           rate += 1;
142         }
143       }
144       if (counter == 0) {
145         rate += 1;
```

```
146       }
147
148       // lecture 1,2,3
149       // practice 1,4,5
150       //      uint pairNum = 0;
151       //      for (auto i = it; i != it + workHours; ++i) {
152       //        if (i->teacher != nullptr) {
153       //          if ((pairNum == 0 || pairNum < 3) && i->subject->
              getType()) {
154       //            rate += 1;
155       //          } else if (pairNum > 3 && i->subject->getType()) {
156       //            rate += 1;
157       //          }
158       //          ++pairNum;
159       //        }
160       //      }
161     }
162   }
163
164   return lp_combination_coef * rate;
165 }
166
167 double AntSystem::rateStudentsGaps(const Solution &candidate) {
168   double rate = 0;
169   for (const auto &c : candidate.studentsSchadule) {
170     for (auto i = c.begin(); i != c.end(); i += workHours) {
171       //      auto first =
172       //          std::find_if(i, i + workHours, [](const StudentCell
              &current)
173       //          {
174       //            return current.teacher != nullptr;
175       //          });
176       //      auto last = std::find_if(std::make_reverse_iterator(i +
177       //      workHours),
178       //                               std::make_reverse_iterator(i),
179       //                               [](const StudentCell &current)
              {
```

```
180       //                                    return current.teacher !=
             nullptr;
181       //                                  })
182       //                        .base();
183
184       auto first = c.begin();
185       for (auto j = i; j != i + workHours; ++j) {
186         if (j->teacher != nullptr) {
187           first = j;
188           break;
189         }
190       }
191       auto last = c.begin();
192       for (auto j = i + workHours - 1; j != i; --j) {
193         if (j->teacher != nullptr) {
194           last = j;
195           break;
196         }
197       }
198
199       if (first != last && last != c.begin()) {
200         auto dayFirst = i;
201         if (dayFirst->teacher == nullptr) {
202           rate += 1;
203         }
204
205         if ((dayFirst + workHours - 1)->teacher == nullptr) {
206           rate += 1;
207         }
208
209         uint gapCounter = 0;
210         for (auto it = first; it != last; ++it) {
211           if (it->teacher == nullptr) {
212             ++gapCounter;
213           }
214         }
215
```

```
216         if (gapCounter == 0 && std::distance(first, last) > 1) {
217           ++rate;
218         }
219       }
220     }
221   }
222
223   return students_gap_coef * rate;
224 }
225
226 void AntSystem::getIterBest() {
227   std::vector<double> candidatesRate(solutionCandidates.size());
228
229   for (uint i = 0; i < solutionCandidates.size(); ++i) {
230     threads[i] = std::shared_ptr<std::thread>(
231         new std::thread([&candidatesRate, i, this] {
232           candidatesRate[i] = rateCandidate(solutionCandidates[i]);
233         }));
234   }
235
236   for (uint i = 0; i < solutionCandidates.size(); ++i) {
237     threads[i]->join();
238   }
239
240   auto it = std::max_element(candidatesRate.begin(), candidatesRate.
       end());
241
242   if (it != candidatesRate.end()) {
243     iterBest.first =
244         solutionCandidates[std::distance(candidatesRate.begin(), it)
              ];
245     iterBest.second = *it;
246   }
247 }
248
249 void AntSystem::run() {
250   createColonies();
```

```
251
252   allTimeBestIndexes.resize(colonies.size());
253
254   uint threadsNum = colonies.size() * antsNum;
255   threads.resize(threadsNum);
256
257   while (done && fabs(allTimeBest.second - univerData.maxPosibleRate)
          > sigma) {
258     for (auto &c : colonies) {
259       c->evaporatePheromone();
260     }
261
262     general_barrier.setTargetValue(threadsNum);
263     vertex_barrier.setTargetValue(threadsNum);
264
265     auto first = threads.begin();
266     for (auto &c : colonies) {
267       c->construct(first, first + antsNum);
268       first += antsNum;
269     }
270
271     for (auto &th : threads) {
272       th->join();
273     }
274
275     getStudentsSolution();
276     constructTeacherSolution(solutionCandidates);
277
278     getIterBest();
279
280     //    std::cout << iterCounter << ". it: " << iterBest.second <<
            "  "
281     //              << "best: " << allTimeBest.second << "\n";
282
283     if (iterBest.second - allTimeBest.second > sigma) {
284       allTimeBest = iterBest;
285       bestSolutionIterNum = iterCounter;
```

```
286       for (uint i = 0; i < colonies.size(); ++i) {
287         allTimeBestIndexes[i] =
288             colonies[i]->getSolutionIndexSeq(allTimeBest.first.
                antIndex);
289       }
290       done = 10;
291     }
292     --done;
293
294     for (uint i = 0; i < colonies.size(); ++i) {
295       colonies[i]->globalUpdatePheromone(allTimeBestIndexes[i],
296                                          allTimeBest.second);
297     }
298
299     ++iterCounter;
300     plot.push_back(iterBest.second);
301
302     solutionCandidates.resize(0);
303   }
304   //  matplotlibcpp::plot(plot);
305   //  matplotlibcpp::show();
306 }
307
308 Solution AntSystem::getResult() const { return allTimeBest.first; }
309
310 double AntSystem::getSolutionRate() const { return allTimeBest.second
      ; }
311
312 void AntSystem::generateSolutions(std::vector<std::vector<uint>>
      vecIndex,
313                                   uint index,
314                                   std::vector<Solution> &solutions) {
315   if (index == vecIndex.size()) return;
316   do {
317     Solution s;
318     bool validSchadule = true;
319     s.studentsSchadule.resize(univerData.groups.size());
```

```
320
321     for (uint i = 0; i < vecIndex.size(); ++i) {
322       std::vector<std::pair<std::shared_ptr<Vertex>, std::shared_ptr<
             Class>>>
323           cells;
324       for (uint j = 0; j < vecIndex[i].size(); ++j) {
325         auto vert =
326             univerData.groups[i]->getSubjectGraph()->getVertex(
                 vecIndex[i][j]);
327
328         if (vert->teacher == nullptr) {
329           cells.push_back({vert, nullptr});
330           continue;
331         }
332
333         auto currentClass = getClass(vert, s, j);
334         if (currentClass == nullptr) {
335           validSchadule = false;
336           break;
337         }
338
339         cells.push_back({vert, currentClass});
340       }
341       s.setStudentsSchadule(cells, i);
342     }
343
344     if (validSchadule) {
345       solutions.push_back(s);
346     }
347     generateSolutions(vecIndex, index + 1, solutions);
348   }
349
350   while (next_permutation(vecIndex[index].begin(), vecIndex[index].
        end()));
351 }
352
353 bool AntSystem::validateSolutions(const Solution &solution) {
```

```
354   for (uint i = 0; i < workDays * workHours; ++i) {
355     for (uint j = 0; j < solution.studentsSchadule.size(); ++j) {
356       if (solution.studentsSchadule[j][i].teacher != nullptr) {
357         for (uint k = j + 1; k < solution.studentsSchadule.size(); ++
              k) {
358           if (solution.studentsSchadule[k][i].teacher != nullptr &&
359               (solution.studentsSchadule[j][i].teacher ==
360                    solution.studentsSchadule[k][i].teacher ||
361                solution.studentsSchadule[j][i].classroom ==
362                    solution.studentsSchadule[k][i].classroom)) {
363             return false;
364           }
365         }
366       }
367     }
368   }
369   return true;
370 }
371
372 std::shared_ptr<Class> AntSystem::getClass(std::shared_ptr<Vertex> &
      vert,
373                                            const Solution &s, uint j)
                                                {
374   int classIndex;
375   if (vert->subject->getType()) {
376     classIndex = 1;
377   } else if (vert->subject->getLabClass()) {
378     classIndex = 2;
379   } else {
380     classIndex = 0;
381   }
382
383   std::shared_ptr<Class> currentClass = nullptr;
384
385   for (uint k = 0; k < univerData.classes.size(); ++k) {
386     uint sizeCounter = 0;
387     bool isFree = false;
```

```
388
389     while (!isFree && sizeCounter < univerData.classes[classIndex].
            size()) {
390       currentClass = univerData.classes[classIndex][sizeCounter];
391       isFree = true;
392       ++sizeCounter;
393       for (uint i = 0; i < s.studentsSchadule.size(); ++i) {
394         if (!s.studentsSchadule[i].empty() &&
395             s.studentsSchadule[i][j].classroom == currentClass) {
396           isFree = false;
397           break;
398         }
399       }
400     }
401     if (!isFree) {
402       currentClass = nullptr;
403       break;
404     }
405   }
406
407   return currentClass;
408 }
409
410 void AntSystem::brutteForce() {
411   for (auto &g : univerData.groups) {
412     while (g->getSubjectGraphSize() < workDays * workHours) {
413       g->getSubjectGraph()->addVertex(std::shared_ptr<Vertex>(new
              Vertex));
414     }
415   }
416
417   std::vector<std::vector<uint>> vertexIndexes(univerData.groups.size
          ());
418   for (auto &v : vertexIndexes) {
419     v.resize(workDays * workHours);
420     std::iota(v.begin(), v.end(), 0);
421   }
```

```
422
423   std::vector<Solution> solutions;
424   generateSolutions(vertexIndexes, 0, solutions);
425
426   for (const auto &s : solutions) {
427     std::cout << "SOLUTION\n";
428     for (auto &g : s.studentsSchadule) {
429       std::cout << "GROUP\n";
430       for (auto &c : g) {
431         if (c.teacher != nullptr) {
432           std::cout << c.teacher->getFIO() << " " << c.subject->
                 getTitle()
433                     << " " << c.classroom->getNumber() << std::endl;
434         } else {
435           std::cout << "GAP\n";
436         }
437       }
438     }
439   }
440
441   for (auto it = solutions.begin(); it != solutions.end(); ++it) {
442     if (!validateSolutions(*it)) {
443       solutions.erase(it);
444     }
445   }
446   constructTeacherSolution(solutions);
447
448   auto bestSolution = *std::max_element(
449       solutions.begin(), solutions.end(), [this](auto lhs, auto rhs)
             {
450         return rateCandidate(lhs) < rateCandidate(rhs);
451       });
452
453   //  std::cout << "rate: " << rateCandidate(bestSolution) << "\n\n";
454
455   //  for (uint i = 0; i < bestSolution.studentsSchadule.size(); ++i)
       {
```

```
456   //    std::cout << univerData.groups[i]->getIndex() << std::endl;
457
458   //    for (auto &g : bestSolution.studentsSchadule[i]) {
459   //      if (g.teacher != nullptr) {
460   //        std::cout << g.teacher->getFIO() << " " << g.subject->
          getTitle() <<
461   //        " "
462   //                  << g.classroom->getNumber() << std::endl;
463   //      } else {
464   //        std::cout << "GAP\n";
465   //      }
466   //    }
467   //  }
468
469   //  std::cout << "\n\n";
470   for (uint i = 0; i < bestSolution.teacherSchadule.size(); ++i) {
471     std::cout << univerData.teachers[i]->getFIO() << std::endl;
472     for (auto &s : bestSolution.teacherSchadule[i]) {
473       if (s.group != nullptr) {
474         std::cout << s.group->getIndex() << " " << s.subject->
              getTitle() << " "
475                   << s.classroom->getNumber() << std::endl;
476       } else {
477         std::cout << "GAP\n";
478       }
479     }
480   }
481 }
482
483 void AntSystem::dummyTest() {
484   std::cout << "ITERATION " << iterCounter << "\n";
485   for (auto &s : solutionCandidates) {
486     uint i = 0;
487     for (auto &st : s.studentsSchadule) {
488       std::cout << "group " << univerData.groups[i]->getIndex() << "\
            n";
489       for (auto &k : st) {
```

```
490         if (k.teacher != nullptr) {
491           std::cout << k.subject->getTitle() << " " << k.teacher->
                getFIO()
492                     << " " << k.classroom->getNumber() << std::endl;
493         } else {
494           std::cout << "gap\n";
495         }
496       }
497       std::cout << std::endl << std::endl;
498     }
499     ++i;
500   }
501
502   for (auto &s : solutionCandidates) {
503     uint i = 0;
504     for (auto &st : s.teacherSchadule) {
505       std::cout << univerData.teachers[i]->getFIO() << "\n";
506       for (auto &k : st) {
507         if (k.group != nullptr) {
508           std::cout << k.group->getIndex() << std::endl;
509         } else {
510           std::cout << "gap\n";
511         }
512       }
513       std::cout << std::endl << std::endl;
514       ++i;
515     }
516   }
517 }
518
519 void AntSystem::printMatrix() {
520   for (const auto &c : colonies) {
521     for (const auto &m : c->pheromone) {
522       for (auto v : m) {
523         std::cout << std::setw(5) << std::setprecision(3) << std::
              left << v
524                   << " ";
```

```
525       }
526       std::cout << "\n";
527     }
528     std::cout << "\n\n";
529   }
530 }
531
532 void AntSystem::dummyPrint() {
533   //  for (auto& s : candidateSolutions) {
534   for (uint i = 0; i < allTimeBest.first.studentsSchadule.size(); ++i
        ) {
535     std::cout << univerData.groups[i]->getIndex() << " variant\n";
536     for (auto &k : allTimeBest.first.studentsSchadule[i]) {
537       if (k.teacher != nullptr) {
538         std::cout << k.subject->getTitle() << " " << k.teacher->
              getFIO() << " "
539                   << k.classroom->getNumber() << std::endl;
540       } else {
541         std::cout << "gap\n";
542       }
543     }
544     std::cout << std::endl;
545   }
546   //  }
547
548   //  for (auto& s : candidateSolutions) {
549   std::cout << "Candidate solution\n";
550   for (auto i = 0u; i < univerData.teachers.size(); ++i) {
551     std::cout << "teacher: " << univerData.teachers[i]->getFIO() <<
          std::endl;
552     for (auto &k : allTimeBest.first.teacherSchadule[i]) {
553       if (k.group != nullptr) {
554         std::cout << k.subject->getTitle() << " " << k.group->
              getIndex() << " "
555                   << k.classroom->getNumber() << std::endl;
556       } else {
557         std::cout << "gap\n";
```

```
558       }
559     }
560     std::cout << std::endl << std::endl;
561   }
562 }
563
564 void AntSystem::printStudentsSchadule(const Solution &solution) {
565   std::cout << "STUDENTS SCHADULE\n\n";
566   for (uint k = 0; k < solution.studentsSchadule.size(); ++k) {
567     std::cout << univerData.groups[k]->getIndex() << std::endl;
568     for (uint i = 0; i < workDays; ++i) {
569       for (uint j = i; j < solution.studentsSchadule[k].size();
570            j += workHours) {
571         if (solution.studentsSchadule[k][j].teacher != nullptr) {
572           std::cout << std::setw(5) << std::left
573                     << solution.studentsSchadule[k][j].subject->
                        getTitle()
574                     << " *"
575                     << solution.studentsSchadule[k][j].subject->
                        getType() << " "
576                     << solution.studentsSchadule[k][j].teacher->
                        getFIO() << " "
577                     << solution.studentsSchadule[k][j].classroom->
                        getNumber()
578                     << "   |\t";
579         } else {
580           std::cout << std::setw(15) << std::left << "GAP"
581                     << "|\t";
582         }
583       }
584       std::cout << std::endl;
585     }
586     std::cout << "\n\n";
587   }
588 }
589
590 void AntSystem::printTeachersSchadule(const Solution &solution) {
```

```
591   std::cout << "TEACHERS SCHADULE\n\n";
592   for (uint k = 0; k < solution.teacherSchadule.size(); ++k) {
593     std::cout << univerData.teachers[k]->getFIO() << std::endl;
594     for (uint i = 0; i < workDays; ++i) {
595       for (uint j = i; j < solution.teacherSchadule[k].size(); j +=
             workHours) {
596         if (solution.teacherSchadule[k][j].group != nullptr) {
597           std::cout << solution.teacherSchadule[k][j].group->getIndex
                 () << " "
598                     << std::setw(5) << std::left
599                     << solution.teacherSchadule[k][j].subject->
                        getTitle() << " "
600                     << solution.teacherSchadule[k][j].classroom->
                        getNumber()
601                     << "  |\t";
602         } else {
603           std::cout << std::setw(15) << std::left << "GAP"
604                     << "|\t";
605         }
606       }
607       std::cout << std::endl;
608     }
609     std::cout << "\n\n";
610   }
611 }
612
613 AntSystem::~AntSystem() {}
```

```
1 #ifndef CLASS_H
2 #define CLASS_H
3 #include <memory>
4 #include <mutex>
5 #include <vector>
6
7 #include "Common.h"
8
9 // special equipment can be different types like phisics or chemistry
    , so  here
```

```
10 // ENUM of different predefined types can be used in future
11
12 class Class {
13   unsigned number;
14   unsigned buildingNumber;
15   unsigned capacity;
16   bool specialEquipment;
17   std::vector<std::shared_ptr<std::mutex>> isFree;
18
19  public:
20   Class(unsigned num = 0, unsigned buildNum = 0, unsigned cap = 0,
21         bool specEq = false)
22       : number(num),
23         buildingNumber(buildNum),
24         capacity(cap),
25         specialEquipment(specEq) {
26     initMutexes(antsNum);
27   }
28
29   unsigned getBuildingNumber() const { return buildingNumber; }
30   void setBuildingNumber(unsigned value) { buildingNumber = value; }
31
32   unsigned getNumber() const { return number; }
33   void setNumber(unsigned value) { number = value; }
34
35   bool getSpecialEquipment() const { return specialEquipment; }
36   void setSpecialEquipment(bool value) { specialEquipment = value; }
37
38   unsigned getCapacity() const { return capacity; }
39   void setCapacity(const unsigned& value) { capacity = value; }
40
41   void initMutexes(int antsNum) {
42     if (isFree.empty()) {
43       isFree.resize(antsNum);
44
45       for (auto& mutex : isFree) {
46         mutex = std::make_shared<std::mutex>();
```

```
47        }
48      }
49    }
50
51    bool try_get(int index) { return isFree[index]->try_lock(); }
52    void release(int index) { isFree[index]->unlock(); }
53
54    ~Class() {}
55  };
56
57  #endif  // CLASS_H
```

```
 1  #ifndef COMMON_H
 2  #define COMMON_H
 3  #include <vector>
 4
 5  // stucture usings
 6  using matrix = std::vector<std::vector<double>>;
 7  const unsigned workHours = 5;
 8  const unsigned workDays = 5;
 9
10  // ACS constants
11  // const double rho_g = 0.2;  // global evaporation rate
12  const double teta0 = 10;  // value with wich the pheromone matrix
       init
13  const double alpha = 2;   // 3 weight of the pheromone trace
14  const double beta = 3;    // 2 weight of heuristic
15  const double q0 = 0.7;    // choise probability
16  const double min_pheromone_value = 9.;
17  const double max_pheromone_value = 15.;
18  const double rho = teta0 / 1000;  // evaporation rate
19  const unsigned antsNum = 4;
20  const double ksi = 1 / antsNum;  // local pheromone increase coef
21  const double ksi_g = 2 * ksi;    // global pheromone increase coef
22
23  // rate constants
24  const double lp_combination_coef = 0.4;
25  const double students_gap_coef = 0.7;
```

```
26 const double teacher_pref_coef = 0.8;
27 const double lecture_order_coef = 0.35;
28 const double gap_rate = 0.1;
29
30 // algorithm constants
31 const unsigned lecureClassCapacity = 60;
32 const unsigned labClassCapacity = 30;
33
34 #endif  // COMMON_H
```

```
 1 #ifndef BARRIER_H
 2 #define BARRIER_H
 3 #include <condition_variable>
 4 #include <iostream>
 5 #include <mutex>
 6
 7 class barrier {
 8   int wait_count;
 9   int target_wait_count;
10   int m_inter_wait_count = 0;
11
12   std::mutex mtx;
13   std::condition_variable cond_var;
14
15  public:
16   barrier() : wait_count(0) {}
17   barrier(int threads_to_wait_for)
18       : wait_count(0), target_wait_count(threads_to_wait_for) {}
19
20   void wait_once() {
21     std::unique_lock<std::mutex> lk(mtx);
22     ++wait_count;
23     if (wait_count != target_wait_count) {
24       cond_var.wait(
25           lk, [this]() mutable { return wait_count ==
                 target_wait_count; });
26     } else {
27       cond_var.notify_all();
```

```
28      }
29    }
30
31    void setTargetValue(uint target) { target_wait_count = target; }
32
33    void decrementTarget() { --target_wait_count; }
34
35    void wait(bool decrease_check = false) {
36      std::unique_lock<std::mutex> lk(mtx);
37      int const current_wait_cycle = m_inter_wait_count;
38
39      if (decrease_check) {
40        --target_wait_count;
41      } else {
42        ++wait_count;
43      }
44
45      if (wait_count != target_wait_count) {
46        cond_var.wait(lk, [this, current_wait_cycle]() {
47          return m_inter_wait_count != current_wait_cycle;
48        });
49      } else {
50        ++m_inter_wait_count;
51        cond_var.notify_all();
52        wait_count = 0;
53      }
54    }
55  };
56
57  #endif  // BARRIER_H
```

```
1  #ifndef GRAPH_H
2  #define GRAPH_H
3
4  #include <vector>
5
6  #include ”Subject.h”
7  #include ”Teacher.h”
```

```
 8
 9 struct Vertex {
10   std::shared_ptr<Teacher> teacher = nullptr;
11   std::shared_ptr<Subject> subject = nullptr;
12
13   Vertex() {}
14   Vertex(std::shared_ptr<Teacher> t, std::shared_ptr<Subject> s)
15       : teacher(t), subject(s) {}
16 };
17
18 class Graph {
19   std::vector<std::shared_ptr<Vertex>> vertexes;
20
21  public:
22   Graph() {}
23   Graph(const std::vector<std::shared_ptr<Vertex>>& vert) : vertexes(
        vert) {}
24   void addVertex(std::shared_ptr<Vertex> newVertex) {
25     vertexes.push_back(newVertex);
26   }
27
28   void addVertex(std::shared_ptr<Teacher> t, std::shared_ptr<Subject>
         s) {
29     vertexes.push_back(std::shared_ptr<Vertex>(new Vertex(t, s)));
30   }
31   void addVertexes(std::shared_ptr<Teacher> t, std::shared_ptr<
        Subject> s,
32                    int hours_per_week) {
33     for (int i = 0; i < hours_per_week; ++i) {
34       vertexes.push_back(std::shared_ptr<Vertex>(new Vertex(t, s)));
35     }
36   }
37
38   const std::vector<std::shared_ptr<Vertex>>& getVertexes() const {
39     return vertexes;
40   }
41
```

```
42    std::shared_ptr<Vertex>& getVertex(unsigned id) { return vertexes[
         id]; }
43    //  std::shared_ptr<Vertex> getVertex(unsigned id) { return
         vertexes[id]; }
44
45    uint size() const { return vertexes.size(); }
46  };
47
48  #endif  // GRAPH_H
```

```
 1  #ifndef GROUP_H
 2  #define GROUP_H
 3  #include <vector>
 4
 5  #include ”Graph.h”
 6  #include ”Subject.h”
 7  #include ”Teacher.h”
 8
 9  class Group {
10    unsigned cource;
11    unsigned capacity;
12    std::string index = ””;
13
14    std::shared_ptr<Graph> subjectGraph;
15
16   public:
17    Group(int cource, int cap, const std::string& idx,
18          std::shared_ptr<Graph> subjects_graph)
19        : cource(cource),
20          capacity(cap),
21          index(idx),
22          subjectGraph(subjects_graph) {}
23
24    unsigned getCapacity() const { return capacity; }
25    void setCapacity(unsigned value) { capacity = value; }
26
27    unsigned getCource() const { return cource; }
28    void setCource(unsigned value) { cource = value; }
```

```
29
30    std::shared_ptr<Graph> getSubjectGraph() const { return
          subjectGraph; }
31    void setSubjectGraph(std::shared_ptr<Graph> value) { subjectGraph =
          value; }
32
33    std::string getIndex() const { return index; }
34    void setIndex(const std::string& value) { index = value; }
35
36    uint getSubjectGraphSize() const { return subjectGraph.get()->size
          (); }
37 };
38
39 #endif  // GROUP_H
```

```
 1 #ifndef SOLUTION_H
 2 #define SOLUTION_H
 3 #include "Class.h"
 4 #include "Graph.h"
 5 #include "Group.h"
 6
 7 struct TeacherCell {
 8    std::shared_ptr<Class> classroom = nullptr;
 9    std::shared_ptr<Group> group = nullptr;
10    std::shared_ptr<Subject> subject = nullptr;
11
12    TeacherCell() {}
13    TeacherCell(std::shared_ptr<Class> cl, std::shared_ptr<Group> gr,
14                std::shared_ptr<Subject> sb)
15        : classroom(cl), group(gr), subject(sb) {}
16
17    ~TeacherCell() {}
18 };
19
20 struct StudentCell {
21    std::shared_ptr<Class> classroom = nullptr;
22    std::shared_ptr<Teacher> teacher = nullptr;
23    std::shared_ptr<Subject> subject = nullptr;
```

```
24
25    StudentCell() {}
26    StudentCell(std::shared_ptr<Class> cl, std::shared_ptr<Teacher> tr,
27                std::shared_ptr<Subject> sb)
28        : classroom(cl), teacher(tr), subject(sb) {}
29
30    StudentCell(
31        std::pair<std::shared_ptr<Vertex>, std::shared_ptr<Class>>
              solution)
32        : classroom(solution.second),
33          teacher(solution.first->teacher),
34          subject(solution.first->subject) {}
35
36    bool operator==(const StudentCell& another) const {
37      return classroom == another.classroom && teacher == another.
            teacher &&
38             subject == another.subject;
39    }
40
41    ~StudentCell() {}
42  };
43
44  struct Solution {
45    std::vector<std::vector<StudentCell>> studentsSchadule;
46    std::vector<std::vector<TeacherCell>> teacherSchadule;
47    uint antIndex;
48
49    void setStudentsSchadule(
50        const std::vector<std::pair<std::shared_ptr<Vertex>,
51                                    std::shared_ptr<Class>>>&
                                       solutionVec,
52        int groupIndex) {
53      studentsSchadule[groupIndex].resize(solutionVec.size());
54      for (auto i = 0u; i < solutionVec.size(); ++i) {
55        studentsSchadule[groupIndex][i] = StudentCell(solutionVec[i]);
56      }
57    }
```

```
58
59    ~Solution() {}
60  };
61
62  #endif  // SOLUTION_H
```

```
 1  #ifndef SUBJECT_H
 2  #define SUBJECT_H
 3  #include <string>
 4  #include <utility>
 5
 6  class Subject {
 7    std::string title;
 8    bool type = false;       // 0 - lecture, 1 - practice
 9    bool labClass = false;  // 0 - ordinary lecture or practice, 1 -
        labaratory
10
11   public:
12    Subject(const std::string &title = "", bool type = false,
13            bool labClassNeed = false)
14        : title(title), type(type), labClass(labClassNeed){};
15
16    bool getType() const { return type; }
17    void setType(bool value) { type = value; }
18
19    bool getLabClass() const { return labClass; }
20    void setLabClass(bool value) { labClass = value; }
21
22    std::string getTitle() const { return title; }
23    void setTitle(const std::string &value) { title = value; }
24  };
25
26  #endif  // SUBJECT_H
```

```
 1  #ifndef TEACHER_H
 2  #define TEACHER_H
 3  #include <array>
 4  #include <iostream>
```

```
5  #include <memory>
6  #include <mutex>
7  #include <string>
8  #include <vector>
9
10 #include ”Common.h”
11
12 using prefmatrix = std::array<bool, workHours * workDays>;
13
14 class Teacher {
15   std::string FIO = ””;
16   uint hours = 0;
17   prefmatrix preferencesMatrix;
18
19   void initPreferenceMatrix() {
20     std::fill(preferencesMatrix.begin(), preferencesMatrix.end(),
          true);
21   }
22
23  public:
24   std::vector<std::shared_ptr<std::mutex>> isFree;
25
26   Teacher() {
27     initPreferenceMatrix();
28     initMutexes(antsNum);
29   };
30
31   Teacher(const std::string &fio, const prefmatrix &preferMatrix =
        prefmatrix{},
32           uint h = 0)
33       : FIO(fio), hours(h) {
34     if (preferMatrix.empty()) {
35       initPreferenceMatrix();
36     } else {
37       preferencesMatrix = preferMatrix;
38     }
39
```

```
40      initMutexes(antsNum);
41    }
42
43    prefmatrix getPreferencesMatrix() const { return preferencesMatrix;
          }
44
45    bool getPref(uint i) { return preferencesMatrix[i]; }
46
47    void setPreferencesMatrix(const prefmatrix &value) {
48      preferencesMatrix = value;
49    }
50
51    std::string getFIO() const { return FIO; }
52    void setFIO(const std::string &value) { FIO = value; }
53
54    uint getHours() const { return hours; }
55    void setHours(const uint &value) { hours = value; }
56
57    bool try_get(int index) { return isFree[index]->try_lock(); }
58    void release(int index) { isFree[index]->unlock(); }
59
60    void initMutexes(int antsNum) {
61      if (isFree.empty()) {
62        isFree.resize(antsNum);
63
64        for (auto &mutex : isFree) {
65          mutex = std::make_shared<std::mutex>();
66        }
67      }
68    }
69
70    ~Teacher() {}
71  };
72
73  #endif  // TEACHER_H
```

```
1  #ifndef UNIVERSITYDATA_H
2  #define UNIVERSITYDATA_H
```

```
3  #include <vector>
4
5  #include ”Class.h”
6  #include ”Group.h”
7
8  using classes_matrix = std::vector<std::vector<std::shared_ptr<Class
       >>>;
9
10 struct UniversityData {
11   classes_matrix classes;
12   std::vector<std::shared_ptr<Teacher>> teachers;
13   std::vector<std::shared_ptr<Subject>> subjects;
14   std::vector<std::shared_ptr<Group>> groups;
15
16   double maxPosibleRate = 1;
17
18   void getMaxPosibleRate() {
19     for (const auto &t : teachers) {
20       maxPosibleRate += t->getHours();
21     }
22     maxPosibleRate *= teacher_pref_coef;
23
24     // no gaps between classes
25     uint tmp = workDays * groups.size();
26
27     // all gaps is ”in right place” first or last timeslot
28     for (const auto &g : groups) {
29       tmp += workDays * workHours - g->getSubjectGraphSize();
30     }
31     maxPosibleRate += tmp * students_gap_coef;
32
33     // no same classes in one day
34     tmp = workDays * groups.size();
35
36     // all practices or labs located after lectures (or lectures and
          practices
37     // in one day)
```

```
38      for (const auto &g : groups) {
39        tmp += g->getSubjectGraphSize() / 2;
40      }
41      maxPosibleRate += tmp * lp_combination_coef;
42    }
43
44    UniversityData() { getMaxPosibleRate(); }
45
46    UniversityData(classes_matrix &cls,
47                   std::vector<std::shared_ptr<Teacher>> &tch,
48                   std::vector<std::shared_ptr<Subject>> &subjs,
49                   std::vector<std::shared_ptr<Group>> &grs)
50        : classes(cls), teachers(tch), subjects(subjs), groups(grs) {
51      getMaxPosibleRate();
52    }
53  };
54  #endif  // UNIVERSITYDATA_H
```

```
1  #include <QApplication>
2
3  #include ”Presentor.h”
4  #include ”mainwindow.h”
5
6  int main(int argc, char *argv[]) {
7    QApplication a(argc, argv);
8    MainWindow w;
9
10   Presentor pres(&w);
11   w.setWindowTitle(”Scheduler”);
12   w.show();
13   return a.exec();
14 }
```

```
1  #include ”mainwindow.h”
2
3  #include <QComboBox>
4  #include <QDebug>
5  #include <QGraphicsEffect>
```

```
6  #include <QMenu>
7  #include <QMessageBox>
8  #include <QParallelAnimationGroup>
9  #include <QTabWidget>
10 #include <QToolButton>
11
12 #include "prefwidget.h"
13 #include "schedule.h"
14 #include "ui_mainwindow.h"
15
16 MainWindow::MainWindow(QWidget *parent)
17     : QMainWindow(parent), ui(new Ui::MainWindow) {
18   ui->setupUi(this);
19   ui->widget_2->setStyleSheet("background-color: #2A2D34;");
20
21   QImage img(":/icon/sch.png");
22   QPixmap pm;
23   pm = pm.fromImage(
24       img.scaled(90, 110, Qt::KeepAspectRatio, Qt::
          SmoothTransformation));
25
26   ui->label->setPixmap(pm);
27   ui->label->setAlignment(Qt::AlignCenter);
28   ui->label->setStyleSheet("padding: 15px 0px;");
29
30   centralWidget()->layout()->setContentsMargins(0, 0, 0, 0);
31
32   ui->gridLayout->setSpacing(0);
33   ui->stackedWidget->setCurrentIndex(0);
34
35   uint i = 0;
36   while (ui->stackedWidget->count() != 1) {
37     widgets.push_back(ui->stackedWidget->currentWidget());
38     ui->stackedWidget->removeWidget(widgets[i]);
39     ++i;
40   }
41
```

```
42  ui->teachersTable->horizontalHeader()->setSectionResizeMode(
43      QHeaderView::Stretch);
44  ui->subjectsTable->horizontalHeader()->setSectionResizeMode(
45      QHeaderView::Stretch);
46  ui->classesTable->horizontalHeader()->setSectionResizeMode(
47      QHeaderView::Stretch);
48
49  connect(ui->sideBar->addAction("Teachers"), &QAction::toggled, [
       this] {
50    ui->stackedWidget->setCurrentIndex(0);
51    animateSlideAndFade((QWidget *)(ui->teachersTable));
52  });
53  connect(ui->sideBar->addAction("Subjects"), &QAction::toggled, [
       this] {
54    ui->stackedWidget->setCurrentIndex(1);
55    animateSlideAndFade((QWidget *)(ui->subjectsTable));
56  });
57  connect(ui->sideBar->addAction("Groups"), &QAction::toggled, [this]
       {
58    ui->stackedWidget->setCurrentIndex(2);
59    animateSlideAndFade((QWidget *)(ui->groupsTab));
60  });
61  connect(ui->sideBar->addAction("Classes"), &QAction::toggled, [this
       ] {
62    ui->stackedWidget->setCurrentIndex(3);
63    animateSlideAndFade((QWidget *)(ui->classesTable));
64  });
65  connect(ui->sideBar->addAction("Schedule"), &QAction::toggled, [
       this] {
66    ui->stackedWidget->setCurrentIndex(4);
67    animateSlideAndFade((QWidget *)(ui->tabWidget));
68  });
69  //  ui->stackedWidget->insertWidget(4, new QTabWidget);
70
71  auto teachersPrevButton = new QPushButton();
72  teachersPrevButton->setIcon(QIcon(":/icon/plus.png"));
73  teachersPrevButton->setIconSize(QSize(80, 80));
```

```
74
75  ui->stackedWidget->insertWidget(0, teachersPrevButton);
76  teachersPrevButton->setSizePolicy(QSizePolicy::Expanding,
77                                    QSizePolicy::Expanding);
78
79  auto subjectsPrevButton = new QPushButton;
80  ui->stackedWidget->insertWidget(1, subjectsPrevButton);
81  subjectsPrevButton->setSizePolicy(QSizePolicy::Expanding,
82                                    QSizePolicy::Expanding);
83
84  subjectsPrevButton->setIcon(QIcon(":/icon/plus.png"));
85  subjectsPrevButton->setIconSize(QSize(80, 80));
86
87  auto groupsPrevButton = new QPushButton;
88  ui->stackedWidget->insertWidget(2, groupsPrevButton);
89  groupsPrevButton->setSizePolicy(QSizePolicy::Expanding,
90                                  QSizePolicy::Expanding);
91
92  groupsPrevButton->setIcon(QIcon(":/icon/plus.png"));
93  groupsPrevButton->setIconSize(QSize(80, 80));
94
95  auto classesPrevButton = new QPushButton;
96  ui->stackedWidget->insertWidget(3, classesPrevButton);
97  classesPrevButton->setSizePolicy(QSizePolicy::Expanding,
98                                   QSizePolicy::Expanding);
99
100 classesPrevButton->setIcon(QIcon(":/icon/plus.png"));
101 classesPrevButton->setIconSize(QSize(80, 80));
102
103 connect(teachersPrevButton, &QAbstractButton::clicked,
104         [this, teachersPrevButton] {
105           ui->stackedWidget->removeWidget(teachersPrevButton);
106           ui->stackedWidget->insertWidget(0, widgets[0]);
107           ui->stackedWidget->setCurrentIndex(0);
108           animateSlideAndFade((QWidget *)(ui->teachersTable));
109
110           teachersPrevButton->deleteLater();
```

```
111         });
112
113 connect(subjectsPrevButton, &QAbstractButton::clicked,
114         [this, subjectsPrevButton] {
115           ui->stackedWidget->removeWidget(subjectsPrevButton);
116           ui->stackedWidget->insertWidget(1, widgets[1]);
117           ui->stackedWidget->setCurrentIndex(1);
118           animateSlideAndFade((QWidget *)(ui->subjectsTable));
119           subjectsPrevButton->deleteLater();
120         });
121
122 connect(groupsPrevButton, &QAbstractButton::clicked,
123         [this, groupsPrevButton] {
124           ui->stackedWidget->removeWidget(groupsPrevButton);
125           ui->stackedWidget->insertWidget(2, widgets[2]);
126           ui->stackedWidget->setCurrentIndex(2);
127           animateSlideAndFade((QWidget *)(ui->groupsTab));
128
129           groupsPrevButton->deleteLater();
130         });
131
132 connect(classesPrevButton, &QAbstractButton::clicked,
133         [this, classesPrevButton] {
134           ui->stackedWidget->removeWidget(classesPrevButton);
135           ui->stackedWidget->insertWidget(3, widgets[3]);
136           ui->stackedWidget->setCurrentIndex(3);
137           animateSlideAndFade((QWidget *)(ui->classesTable));
138           classesPrevButton->deleteLater();
139         });
140
141 classesPrevButton->setObjectName(”prevBtn”);
142 teachersPrevButton->setObjectName(”prevBtn”);
143 groupsPrevButton->setObjectName(”prevBtn”);
144 subjectsPrevButton->setObjectName(”prevBtn”);
145 teachersPrevButton->setStyleSheet(”margin:0;”);
146
147 ui->sideBar->_checkedAction = ui->sideBar->actions().first();
```

```
148   ui->sideBar->actions().first()->setChecked(true);
149 }
150
151 MainWindow::~MainWindow() { delete ui; }
152
153 QPushButton *MainWindow::addButton() { return ui->addTeacher; }
154
155 QPushButton *MainWindow::addGroupButton() { return ui->addGroupButton
      ; }
156
157 QPushButton *MainWindow::addClassButton() { return ui->addClassButton
      ; }
158
159 QPushButton *MainWindow::addItemButton() { return ui->addItemButton;
      }
160
161 QPushButton *MainWindow::removeItemButton() { return ui->
      removeItemButton; }
162
163 QPushButton *MainWindow::removeGroupButton() { return ui->
      removeGroupButton; }
164
165 QPushButton *MainWindow::removeClassButton() { return ui->
      removeClassButton; }
166
167 QPushButton *MainWindow::addSubject() { return ui->addSubject; }
168
169 QPushButton *MainWindow::removeSubject() { return ui->removeSubject;
      }
170
171 QPushButton *MainWindow::createButton() { return ui->createButton; }
172
173 QPushButton *MainWindow::removeButton() { return ui->removeTeacher; }
174
175 QTableWidget *MainWindow::teachersTable() { return ui->teachersTable;
       }
176
```

```
177 QTableWidget *MainWindow::subjectsTable() { return ui->subjectsTable;
        }
178
179 QTabWidget *MainWindow::tabWidget() { return ui->tabWidget; }
180
181 QTabWidget *MainWindow::groupsTab() { return ui->groupsTab; }
182
183 QTableWidget *MainWindow::classesTable() { return ui->classesTable; }
184
185 void MainWindow::animateSlideAndFade(QWidget *w) {
186   QGraphicsOpacityEffect *eff = new QGraphicsOpacityEffect(this);
187
188   QPropertyAnimation *a = new QPropertyAnimation(eff, "opacity");
189   a->setDuration(190);
190   a->setStartValue(0);
191   a->setEndValue(1);
192   a->setEasingCurve(QEasingCurve::OutBack);
193   w->setGraphicsEffect(eff);
194
195   QPropertyAnimation *a1 = new QPropertyAnimation(w, "geometry");
196
197   a1->setDuration(200);
198   auto pos = w->pos();
199   auto size = w->size();
200   a1->setStartValue(QRect(pos.x(), pos.y() + 100, size.width(), size.
        height()));
201   a1->setEndValue(QRect(pos.x(), pos.y(), size.width(), size.height()
        ));
202
203   a1->start(QPropertyAnimation::DeleteWhenStopped);
204   a->start(QPropertyAnimation::DeleteWhenStopped);
205 }
```

```
1 #ifndef MAINWINDOW_H
2 #define MAINWINDOW_H
3
4 #include <QMainWindow>
5 #include <QPushButton>
```

```
6  #include <QTableWidget>
7
8  #include "model/Teacher.h"
9
10 QT_BEGIN_NAMESPACE
11 namespace Ui {
12 class MainWindow;
13 }
14 QT_END_NAMESPACE
15
16 class MainWindow : public QMainWindow {
17   Q_OBJECT
18
19   QVector<QWidget *> widgets;
20
21  public:
22   MainWindow(QWidget *parent = nullptr);
23   ~MainWindow();
24
25   QPushButton *addButton();
26   QPushButton *addGroupButton();
27   QPushButton *addClassButton();
28   QPushButton *addItemButton();
29   QPushButton *removeItemButton();
30   QPushButton *removeGroupButton();
31   QPushButton *removeClassButton();
32   QPushButton *addSubject();
33   QPushButton *removeSubject();
34   QPushButton *createButton();
35   QPushButton *removeButton();
36   QTableWidget *teachersTable();
37   QTableWidget *subjectsTable();
38   QTabWidget *tabWidget();
39   QTabWidget *groupsTab();
40   QTableWidget *classesTable();
41
42  private:
```

```
43   void animateSlideAndFade(QWidget *w);
44   Ui::MainWindow *ui;
45   QTableWidget *teachsTable;
46   QTableWidget *subjsTable;
47   QTableWidget *grsTable;
48   QVector<prefmatrix> prefs;
49 };
50 #endif  // MAINWINDOW_H
```

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <ui version="4.0">
3   <class>MainWindow</class>
4   <widget class="QMainWindow" name="MainWindow">
5    <property name="geometry">
6     <rect>
7      <x>0</x>
8      <y>0</y>
9      <width>815</width>
10     <height>517</height>
11    </rect>
12   </property>
13   <property name="windowTitle">
14    <string>MainWindow</string>
15   </property>
16   <property name="styleSheet">
17    <string notr="true"/>
18   </property>
19   <widget class="QWidget" name="centralwidget">
20    <property name="styleSheet">
21     <string notr="true">*{
22 font-family: Noto Sans CJK;
23 border: none;
24 }
25
26 QLabel
27 {
28   text-align:center;
29 }
```

```
30
31 centralwidget
32 {
33 background-color: grey;
34 }
35
36 QPushButton:hover{
37 border: 1px solid #2667FF
38 }
39
40 QPushButton#prevBtn:hover{
41 border:  none;
42 }
43
44 QPushButton#prevBtn
45 {
46 margin: 0px;
47 background-color: #EFEFEF;
48 padding-bottom: 20px;
49 }
50
51 QPushButton{
52 background-color: white ;
53 color: #2A2D34;
54 border:none;
55 padding:6px;
56 }
57
58 QTableCornerButton::section
59 {
60 background-color: white ;
61 border: 1px solid #2667FF;
62 border-top:none;
63 border-left:none;
64 border-right: none;
65 }
66
```

```
QComboBox::item:selected{
selection-background-color: white;
selection-color: black;
}

QHeaderView {
    background-color: white;
}

QHeaderView::section {
    background-color: white;
    color: #2A2D34;
    padding: 4px;
    border-style: none;
    border-bottom: 1px solid blue;
    border-right: none;
}

QHeaderView::section:horizontal
{
    border-top: none;
}

QHeaderView::section:vertical
{
    border-left: none;
  border-bottom: 1px solid #c0c0c0;

}

QHeaderView::section:vertical
{
background-color: white;
}

QTableWidget:edit-focus{
selection-background-color: #ffffff;
```

```
104 border: 1px solid #DA7422 ;
105 }
106
107 QTableWidget{
108 border:none;
109 }
110
111 QTableWidget::item {
112 selection-background-color: none;
113 }
114
115 QTableWidget::item:selected {
116 border: 1px solid #2667FF ;
117 background-color: none;
118 }
119
120 QWidget#stackedWidget
121 {
122   margin: 15px;
123 }
124
125 QTabWidget::pane{
126 border:  1px solid white;
127 background-color:white;
128 }
129
130 QTabWidget{
131 border:  2px solid white;
132 background-color:white;
133 }
134
135 QTabBar::tab
136 {
137 border:  2px solid lightgray;
138 background-color: lightgray;
139 padding: 4px;
140 }
```

```
141
142 QTabBar::tab::selected
143 {
144 background-color:white;
145 border:  2px solid white;
146 padding: 4px;
147 }
148
149 QTabWidget#tabWidget::pane
150 {
151   border: 2px solid white;
152
153
154 }
155
156 QScrollBar{
157         border: none;
158         background: none;
159         width: 3px;
160         margin: 0px 0px 0px 0px;
161     alternate-background-color: #D5EAFF;
162     }
163
164
165     QScrollBar::add-line{
166         height: 0px;
167         subcontrol-position: bottom;
168         subcontrol-origin: margin;
169     }
170
171     QScrollBar::sub-line{
172         height: 0 px;
173         subcontrol-position: top;
174         subcontrol-origin: margin;
175     }
176
177
```

```
178  QScrollBar::handle:vertical {
179          background: gray;
180          min-height: 0px;
181      }
182      </string>
183     </property>
184     <layout class="QGridLayout" name="gridLayout">
185      <item row="0" column="1">
186       <widget class="QStackedWidget" name="stackedWidget">
187        <property name="currentIndex">
188         <number>2</number>
189        </property>
190        <widget class="QWidget" name="firstPage">
191         <layout class="QGridLayout" name="gridLayout_2">
192          <item row="1" column="0">
193           <layout class="QVBoxLayout" name="verticalLayout">
194            <property name="spacing">
195             <number>6</number>
196            </property>
197            <item>
198             <widget class="QTableWidget" name="teachersTable">
199              <column>
200               <property name="text">
201                <string>FIO</string>
202               </property>
203              </column>
204              <column>
205               <property name="text">
206                <string>Hours per week</string>
207               </property>
208              </column>
209              <column>
210               <property name="text">
211                <string>Preferences</string>
212               </property>
213              </column>
214             </widget>
```

```
215          </item>
216         </layout>
217        </item>
218        <item row="0" column="0">
219         <layout class="QHBoxLayout" name="firstPageHorizontalLayout
               ">
220          <property name="spacing">
221           <number>20</number>
222          </property>
223          <property name="bottomMargin">
224           <number>15</number>
225          </property>
226          <item>
227           <spacer name="horizontalSpacer">
228            <property name="orientation">
229             <enum>Qt::Horizontal</enum>
230            </property>
231            <property name="sizeHint" stdset="0">
232             <size>
233              <width>40</width>
234              <height>20</height>
235             </size>
236            </property>
237           </spacer>
238          </item>
239          <item>
240           <widget class="QPushButton" name="removeTeacher">
241            <property name="text">
242             <string>Remove</string>
243            </property>
244            <property name="icon">
245             <iconset resource="icon.qrc">
246              <normaloff>:/icon/003-trash-can.png</normaloff>:/icon
                    /003-trash-can.png</iconset>
247            </property>
248           </widget>
249          </item>
```

```
250          <item>
251           <widget class="QPushButton" name="addTeacher">
252            <property name="text">
253             <string>Add</string>
254            </property>
255            <property name="icon">
256             <iconset resource="icon.qrc">
257              <normaloff>:/icon/002-plus.png</normaloff>:/icon/002-
                    plus.png</iconset>
258            </property>
259           </widget>
260          </item>
261         </layout>
262        </item>
263       </layout>
264      </widget>
265      <widget class="QWidget" name="secondPage">
266       <layout class="QGridLayout" name="gridLayout_4">
267        <item row="0" column="0">
268         <layout class="QVBoxLayout" name="verticalLayout_3">
269          <item>
270           <layout class="QHBoxLayout" name="horizontalLayout_3">
271            <property name="spacing">
272             <number>20</number>
273            </property>
274            <property name="bottomMargin">
275             <number>15</number>
276            </property>
277            <item>
278             <spacer name="horizontalSpacer_3">
279              <property name="orientation">
280               <enum>Qt::Horizontal</enum>
281              </property>
282              <property name="sizeHint" stdset="0">
283               <size>
284                <width>40</width>
285                <height>20</height>
```

```
286                 </size>
287                </property>
288               </spacer>
289              </item>
290              <item>
291               <widget class="QPushButton" name="removeSubject">
292                <property name="text">
293                 <string>Remove</string>
294                </property>
295                <property name="icon">
296                 <iconset resource="icon.qrc">
297                  <normaloff>:/icon/003-trash-can.png</normaloff>:/icon
                        /003-trash-can.png</iconset>
298                </property>
299               </widget>
300              </item>
301              <item>
302               <widget class="QPushButton" name="addSubject">
303                <property name="text">
304                 <string>Add</string>
305                </property>
306                <property name="icon">
307                 <iconset resource="icon.qrc">
308                  <normaloff>:/icon/002-plus.png</normaloff>:/icon/002-
                        plus.png</iconset>
309                </property>
310               </widget>
311              </item>
312             </layout>
313            </item>
314            <item>
315             <widget class="QTableWidget" name="subjectsTable">
316              <column>
317               <property name="text">
318                <string>Title</string>
319               </property>
320              </column>
```

```
321             <column>
322              <property name="text">
323               <string>Subject type</string>
324              </property>
325             </column>
326             <column>
327              <property name="text">
328               <string>Special class</string>
329              </property>
330             </column>
331            </widget>
332           </item>
333          </layout>
334         </item>
335        </layout>
336       </widget>
337       <widget class="QWidget" name="groupsPage">
338        <layout class="QGridLayout" name="gridLayout_5">
339         <item row="1" column="0">
340          <widget class="QTabWidget" name="groupsTab">
341           <property name="styleSheet">
342            <string notr="true">border: 1px solid white;</string>
343           </property>
344          </widget>
345         </item>
346         <item row="0" column="0">
347          <layout class="QHBoxLayout" name="horizontalLayout_4">
348           <property name="spacing">
349            <number>20</number>
350           </property>
351           <property name="bottomMargin">
352            <number>15</number>
353           </property>
354           <item>
355            <widget class="QPushButton" name="addGroupButton">
356             <property name="text">
357              <string>Add group</string>
```

```
358             </property>
359             <property name="icon">
360              <iconset resource="icon.qrc">
361               <normaloff>:/icon/002-plus.png</normaloff>:/icon/002-
                     plus.png</iconset>
362             </property>
363            </widget>
364           </item>
365           <item>
366            <widget class="QPushButton" name="removeGroupButton">
367             <property name="text">
368              <string>Remove group</string>
369             </property>
370             <property name="icon">
371              <iconset resource="icon.qrc">
372               <normaloff>:/icon/003-trash-can.png</normaloff>:/icon
                     /003-trash-can.png</iconset>
373             </property>
374            </widget>
375           </item>
376           <item>
377            <spacer name="horizontalSpacer_4">
378             <property name="orientation">
379              <enum>Qt::Horizontal</enum>
380             </property>
381             <property name="sizeHint" stdset="0">
382              <size>
383               <width>40</width>
384               <height>20</height>
385              </size>
386             </property>
387            </spacer>
388           </item>
389           <item>
390            <widget class="QPushButton" name="addItemButton">
391             <property name="font">
392              <font>
```

```
393              <family>Noto Sans CJK</family>
394              <weight>50</weight>
395              <bold>false</bold>
396             </font>
397            </property>
398            <property name="text">
399             <string>Add item</string>
400            </property>
401            <property name="icon">
402             <iconset resource="icon.qrc">
403              <normaloff>:/icon/002-plus.png</normaloff>:/icon/002-
                    plus.png</iconset>
404            </property>
405           </widget>
406          </item>
407          <item>
408           <widget class="QPushButton" name="removeItemButton">
409            <property name="text">
410             <string>Remove item</string>
411            </property>
412            <property name="icon">
413             <iconset resource="icon.qrc">
414              <normaloff>:/icon/003-trash-can.png</normaloff>:/icon
                    /003-trash-can.png</iconset>
415            </property>
416           </widget>
417          </item>
418         </layout>
419        </item>
420       </layout>
421      </widget>
422      <widget class="QWidget" name="classsesPage">
423       <layout class="QGridLayout" name="gridLayout_8">
424        <property name="leftMargin">
425         <number>9</number>
426        </property>
427        <property name="topMargin">
```

```
428          <number>9</number>
429         </property>
430         <property name="rightMargin">
431          <number>9</number>
432         </property>
433         <property name="bottomMargin">
434          <number>9</number>
435         </property>
436         <item row="0" column="0">
437          <layout class="QVBoxLayout" name="verticalLayout_5">
438           <item>
439            <layout class="QHBoxLayout" name="horizontalLayout_6">
440             <property name="spacing">
441              <number>20</number>
442             </property>
443             <property name="bottomMargin">
444              <number>15</number>
445             </property>
446             <item>
447              <spacer name="horizontalSpacer_6">
448               <property name="orientation">
449                <enum>Qt::Horizontal</enum>
450               </property>
451               <property name="sizeHint" stdset="0">
452                <size>
453                 <width>40</width>
454                 <height>20</height>
455                </size>
456               </property>
457              </spacer>
458             </item>
459             <item>
460              <widget class="QPushButton" name="removeClassButton">
461               <property name="styleSheet">
462                <string notr="true"/>
463               </property>
464               <property name="text">
```

```
465                <string>Remove</string>
466               </property>
467               <property name="icon">
468                <iconset resource="icon.qrc">
469                 <normaloff>:/icon/003-trash-can.png</normaloff>:/icon
                       /003-trash-can.png</iconset>
470               </property>
471              </widget>
472             </item>
473             <item>
474              <widget class="QPushButton" name="addClassButton">
475               <property name="styleSheet">
476                <string notr="true"/>
477               </property>
478               <property name="text">
479                <string>Add</string>
480               </property>
481               <property name="icon">
482                <iconset resource="icon.qrc">
483                 <normaloff>:/icon/002-plus.png</normaloff>:/icon/002-
                       plus.png</iconset>
484               </property>
485              </widget>
486             </item>
487            </layout>
488           </item>
489           <item>
490            <widget class="QTableWidget" name="classesTable">
491             <column>
492              <property name="text">
493               <string>Class number</string>
494              </property>
495             </column>
496             <column>
497              <property name="text">
498               <string>Building number</string>
499              </property>
```

```
500             </column>
501             <column>
502              <property name="text">
503               <string>Capacity</string>
504              </property>
505             </column>
506             <column>
507              <property name="text">
508               <string>Special class</string>
509              </property>
510             </column>
511            </widget>
512           </item>
513          </layout>
514         </item>
515        </layout>
516       </widget>
517       <widget class="QWidget" name="schedulePage">
518        <layout class="QGridLayout" name="gridLayout_3">
519         <item row="0" column="0">
520          <layout class="QVBoxLayout" name="verticalLayout_2">
521           <property name="spacing">
522            <number>6</number>
523           </property>
524           <item>
525            <widget class="QTabWidget" name="tabWidget">
526             <property name="currentIndex">
527              <number>-1</number>
528             </property>
529            </widget>
530           </item>
531           <item>
532            <layout class="QHBoxLayout" name="horizontalLayout_2">
533             <property name="spacing">
534              <number>0</number>
535             </property>
536             <item>
```

```
537              <spacer name="horizontalSpacer_2">
538               <property name="orientation">
539                <enum>Qt::Horizontal</enum>
540               </property>
541               <property name="sizeHint" stdset="0">
542                <size>
543                 <width>40</width>
544                 <height>20</height>
545                </size>
546               </property>
547              </spacer>
548             </item>
549             <item>
550              <widget class="QPushButton" name="createButton">
551               <property name="text">
552                <string>Create</string>
553               </property>
554               <property name="icon">
555                <iconset resource="icon.qrc">
556                 <normaloff>:/icon/001-time-management.png</normaloff
                        >:/icon/001-time-management.png</iconset>
557               </property>
558              </widget>
559             </item>
560            </layout>
561           </item>
562          </layout>
563         </item>
564        </layout>
565       </widget>
566      </widget>
567     </item>
568     <item row="0" column="0">
569      <widget class="QWidget" name="widget_2" native="true">
570       <property name="styleSheet">
571        <string notr="true"/>
572       </property>
```

```
573        <layout class="QVBoxLayout" name="verticalLayout_4">
574         <property name="leftMargin">
575          <number>0</number>
576         </property>
577         <property name="topMargin">
578          <number>0</number>
579         </property>
580         <property name="rightMargin">
581          <number>0</number>
582         </property>
583         <property name="bottomMargin">
584          <number>0</number>
585         </property>
586         <item>
587          <widget class="QLabel" name="label">
588           <property name="text">
589            <string/>
590           </property>
591           <property name="alignment">
592            <set>Qt::AlignJustify|Qt::AlignVCenter</set>
593           </property>
594          </widget>
595         </item>
596         <item>
597          <widget class="SideBar" name="sideBar" native="true">
598           <property name="styleSheet">
599            <string notr="true"/>
600           </property>
601          </widget>
602         </item>
603         <item>
604          <spacer name="verticalSpacer">
605           <property name="orientation">
606            <enum>Qt::Vertical</enum>
607           </property>
608           <property name="sizeHint" stdset="0">
609            <size>
```

```
610             <width>17</width>
611             <height>560</height>
612            </size>
613           </property>
614          </spacer>
615         </item>
616        </layout>
617       </widget>
618      </item>
619     </layout>
620    </widget>
621   </widget>
622   <customwidgets>
623    <customwidget>
624     <class>SideBar</class>
625     <extends>QWidget</extends>
626     <header>sidebar.h</header>
627     <container>1</container>
628    </customwidget>
629   </customwidgets>
630   <resources>
631    <include location="icon.qrc"/>
632   </resources>
633   <connections/>
634 </ui>
```

```
1  #ifndef PREFWIDGET_H
2  #define PREFWIDGET_H
3
4  #include <QVector>
5  #include <QWidget>
6
7  #include "model/UniversityData.h"
8
9  namespace Ui {
10 class PrefWidget;
11 }
12
```

```
13 class PrefWidget : public QWidget {
14   Q_OBJECT
15
16  public:
17   explicit PrefWidget(QVector<prefmatrix> &prefs, uint currentIndex,
18                       QWidget *parent = nullptr);
19   ~PrefWidget();
20
21  private:
22   Ui::PrefWidget *ui;
23
24  signals:
25   void closedWithoutChanges();
26 };
27
28 #endif  // PREFWIDGET_H
```

```
 1 #include "prefwidget.h"
 2
 3 #include <QCloseEvent>
 4 #include <QColor>
 5 #include <QDebug>
 6 #include <QVector>
 7 #include <array>
 8
 9 #include "ui_prefwidget.h"
10
11 PrefWidget::PrefWidget(QVector<prefmatrix> &prefs, uint currentIndex,
12                        QWidget *parent)
13     : QWidget(parent), ui(new Ui::PrefWidget) {
14   ui->setupUi(this);
15   ui->tableWidget->setEditTriggers(QAbstractItemView::NoEditTriggers)
       ;
16
17   for (int i = 0; i < ui->tableWidget->rowCount(); ++i) {
18     for (int j = 0; j < ui->tableWidget->columnCount(); ++j) {
19       auto btn = new QPushButton;
20       btn->setCheckable(true);
```

```
21        connect(btn, &QPushButton::toggled, [btn](bool togled) {
22          if (togled)
23            btn->setStyleSheet("background-color: #2667FF;");
24          else
25            btn->setStyleSheet("");
26        });
27        ui->tableWidget->setCellWidget(i, j, btn);
28      }
29    }
30
31    QObject::connect(
32        ui->saveButton, &QAbstractButton::clicked, [this, &prefs,
             currentIndex] {
33          for (int j = 0; j < ui->tableWidget->columnCount(); ++j) {
34            for (int i = 0; i < ui->tableWidget->rowCount(); ++i) {
35              if (qobject_cast<QPushButton *>(ui->tableWidget->
                    cellWidget(i, j))
36                        ->isChecked()) {
37                prefs[currentIndex][i + j] = true;
38              }
39            }
40          }
41          close();
42        });
43
44    QObject::connect(ui->cancelButton, &QAbstractButton::clicked, [this
        ] {
45      emit closedWithoutChanges();
46      close();
47    });
48  }
49
50  PrefWidget::~PrefWidget() { delete ui; }
```

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <ui version="4.0">
3   <class>PrefWidget</class>
4   <widget class="QWidget" name="PrefWidget">
```

```
5    <property name="geometry">
6     <rect>
7      <x>0</x>
8      <y>0</y>
9      <width>539</width>
10     <height>228</height>
11    </rect>
12   </property>
13   <property name="minimumSize">
14    <size>
15     <width>539</width>
16     <height>228</height>
17    </size>
18   </property>
19   <property name="maximumSize">
20    <size>
21     <width>539</width>
22     <height>228</height>
23    </size>
24   </property>
25   <property name="windowTitle">
26    <string>Form</string>
27   </property>
28   <property name="styleSheet">
29    <string notr="true">QTableWidget:edit-focus{
30 selection-background-color: #ffffff;
31 }
32
33 QTableWidget::item:selected {
34 background-color: #ffffff;
35 }
36
37 </string>
38   </property>
39   <layout class="QGridLayout" name="gridLayout">
40    <item row="0" column="0">
41     <widget class="QTableWidget" name="tableWidget">
```

```
42       <property name="styleSheet">
43        <string notr="true">QPushButton
44 {
45 border: 0px solid black;
46 }</string>
47       </property>
48       <row>
49        <property name="text">
50         <string>1</string>
51        </property>
52       </row>
53       <row>
54        <property name="text">
55         <string>2</string>
56        </property>
57       </row>
58       <row>
59        <property name="text">
60         <string>3</string>
61        </property>
62       </row>
63       <row>
64        <property name="text">
65         <string>4</string>
66        </property>
67       </row>
68       <row>
69        <property name="text">
70         <string>5</string>
71        </property>
72       </row>
73       <column>
74        <property name="text">
75         <string>Monday</string>
76        </property>
77       </column>
78       <column>
```

```
79        <property name="text">
80         <string>Tuesday</string>
81        </property>
82       </column>
83       <column>
84        <property name="text">
85         <string>Wednesday</string>
86        </property>
87       </column>
88       <column>
89        <property name="text">
90         <string>Thursday</string>
91        </property>
92       </column>
93       <column>
94        <property name="text">
95         <string>Friday</string>
96        </property>
97       </column>
98      </widget>
99     </item>
100    <item row="1" column="0">
101     <layout class="QHBoxLayout" name="horizontalLayout">
102      <item>
103       <widget class="QPushButton" name="cancelButton">
104        <property name="text">
105         <string>Cancel</string>
106        </property>
107       </widget>
108      </item>
109      <item>
110       <spacer name="horizontalSpacer">
111        <property name="orientation">
112         <enum>Qt::Horizontal</enum>
113        </property>
114        <property name="sizeHint" stdset="0">
115         <size>
```

```
          <width>40</width>
          <height>20</height>
         </size>
        </property>
       </spacer>
      </item>
      <item>
       <widget class="QPushButton" name="saveButton">
        <property name="text">
         <string>Save</string>
        </property>
       </widget>
      </item>
     </layout>
    </item>
   </layout>
  </widget>
  <resources/>
  <connections/>
</ui>
```

```
#ifndef SCHADULETABLE_H
#define SCHADULETABLE_H

#include <QWidget>
struct item {
  QString subj = "";
  QString subjType = "";
  QString teacher = "";
  QString classNum = "";
};
namespace Ui {
class schaduleTable;
}

class schaduleTable : public QWidget {
  Q_OBJECT

```

```
18  public:
19   explicit schaduleTable(QWidget *parent = nullptr);
20   void addColumn(const QString &title, const QVector<item> &
        columnData);
21   ~schaduleTable();
22
23  private:
24   Ui::schaduleTable *ui;
25 };
26
27 #endif  // SCHADULETABLE_H
```

```
1  #include ”schaduletable.h”
2
3  #include ”ui_schaduletable.h”
4  #include ”verticallabel.h”
5
6  schaduleTable::schaduleTable(QWidget *parent)
7      : QWidget(parent), ui(new Ui::schaduleTable) {
8    ui->setupUi(this);
9
10   const uint grSize = 3;
11   const uint rowsize = 25;
12   ui->tableWidget->setColumnCount(2);
13   ui->tableWidget->setRowCount(rowsize);
14   for (uint i = 0; i < rowsize; i += 5) {
15     ui->tableWidget->setSpan(i, 0, 5, 1);
16   }
17
18   ui->tableWidget->setColumnWidth(0, 30);
19   ui->tableWidget->setColumnWidth(1, 20);
20   for (int i = 2; i < ui->tableWidget->columnCount(); ++i) {
21     ui->tableWidget->setColumnWidth(i, (800 - 50) / grSize);
22   }
23
24   ui->tableWidget->verticalHeader()->hide();
25
```

```
26    QVector<QString> days = {"Monday", "Tuesday", "Wednesday", "
         Thursday",
27                             "Friday"};
28
29    uint counter = 0;
30    for (uint i = 0; i < 25; ++i) {
31      ui->tableWidget->setCellWidget(
32          i, 1, new QLabel(QString::number(counter % 5 + 1)));
33      ++counter;
34    }
35    for (int i = 0; i < 5; ++i) {
36      ui->tableWidget->setCellWidget(i * 5, 0, new VerticalLabel(days[i
           ]));
37    }
38  }
39
40  void schaduleTable::addColumn(const QString &title,
41                                const QVector<item> &columnData) {
42    ui->tableWidget->insertColumn(ui->tableWidget->columnCount());
43    for (int i = 0; i < columnData.size(); ++i) {
44      QLabel *item = new QLabel;
45      item->setTextFormat(Qt::RichText);
46      item->setAlignment(Qt::AlignCenter);
47      item->setStyleSheet("font-size:15pt; padding:5px; ");
48      item->setText(
49          QString("<strong>%1</strong><sub><br>%2<\br></sub><sub><br
              >%3<\br></"
50                  "sub><sub><br>%4<\br></sub>")
51              .arg(columnData[i].subj)
52              .arg(columnData[i].subjType)
53              .arg(columnData[i].teacher)
54              .arg(columnData[i].classNum));
55      ui->tableWidget->setCellWidget(i, ui->tableWidget->columnCount()
           - 1, item);
56      ui->tableWidget->setRowHeight(i, item->height());
57    }
```

```
58   ui->tableWidget->setHorizontalHeaderItem(ui->tableWidget->
        columnCount() - 1,
59                                                new QTableWidgetItem(title
                                                     ));
60   ui->tableWidget->setColumnWidth(ui->tableWidget->columnCount() - 1,
         300);
61   //  ui->tableWidget->horizontalHeader()->stretchLastSection();
62 }
63
64 schaduleTable::~schaduleTable() { delete ui; }
```

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <ui version="4.0">
3   <class>schaduleTable</class>
4   <widget class="QWidget" name="schaduleTable">
5    <property name="geometry">
6     <rect>
7      <x>0</x>
8      <y>0</y>
9      <width>725</width>
10     <height>638</height>
11    </rect>
12   </property>
13   <property name="windowTitle">
14    <string>Form</string>
15   </property>
16   <layout class="QGridLayout" name="gridLayout">
17    <item row="0" column="0">
18     <widget class="QTableWidget" name="tableWidget"/>
19    </item>
20   </layout>
21  </widget>
22  <resources/>
23  <connections/>
24 </ui>
```

```
1 #include "schedule.h"
2
```

```
3  #include <QGraphicsEffect>
4  #include <QMouseEvent>
5  #include <QVariant>
6
7  void Schedule::leftClickAnimation(elem *w, int value, bool reset) {
8    QPropertyAnimation *a2 = new QPropertyAnimation(w, "geometry");
9
10   a2->setDuration(100);
11   auto pos = w->pos();
12   auto size = w->size();
13   a2->setStartValue(QRect(pos.x(), pos.y(), size.width(), size.height
        ()));
14   a2->setEndValue(QRect((pos.x() - value / 2), (pos.y() - value / 2),
15                         size.width() + value, size.height() + value))
                             ;
16
17   if (!reset) {
18     a2->setEasingCurve(QEasingCurve::InCubic);
19     if (w->type == 0) {
20       w->setStyleSheet("border: 2px solid #FFC060;" + w->lectureStyle
           );
21     } else if (w->type == 1) {
22       w->setStyleSheet(w->practiceStyle + "border: 2px solid #8043E8
           ");
23
24     } else if (w->type == 2) {
25       w->setStyleSheet(w->labStyle + "border: 2px solid #FF5D74 ");
26     }
27
28   } else {
29     a2->setEasingCurve(QEasingCurve::OutCubic);
30     if (w->type == 0) {
31       w->setStyleSheet(w->lectureStyle);
32     } else if (w->type == 1) {
33       w->setStyleSheet(w->practiceStyle);
34
35     } else if (w->type == 2) {
```

```
36        w->setStyleSheet(w->labStyle);
37      }
38    }
39    w->raise();
40    a2->start(QPropertyAnimation::DeleteWhenStopped);
41  }
42
43  uint Schedule::getNum() const { return num; }
44
45  void Schedule::setNum(const uint &value) { num = value; }
46
47  bool Schedule::getTeacher_schedule_flag() const {
48    return teacher_schedule_flag;
49  }
50
51  void Schedule::setTeacher_schedule_flag(bool value) {
52    teacher_schedule_flag = value;
53  }
54
55  Schedule::Schedule(QWidget *parent) : QScrollArea(parent) {
56    content = new QWidget(this);
57
58    layout = new QGridLayout(this);
59    //  layout->setHorizontalSpacing(0);
60    //  layout->setVerticalSpacing(10);
61    //  layout->setContentsMargins(2, 2, 2, 2);
62    layout->setSpacing(5);
63    QString title = "Group%1";
64    QString temp = "Teacher";
65    qDebug() << teacher_schedule_flag;
66    if (!teacher_schedule_flag == 1) {
67      title = "Teacher%1";
68      temp = "Group";
69    }
70
71    for (auto i = 0u; i < 5; ++i) {
72      auto gr = new QLabel(i ? title.arg(i) : "");
```

```
73      gr->setAlignment(Qt::AlignCenter);
74      gr->setStyleSheet("font: 12pt bold; background-color:white;
           padding: 0px");
75      layout->addWidget(gr, 0, i);
76
77      QString style("border: none;  background-color:none; text-align:
           center;");
78      for (auto j = 1u; j < 5; ++j) {
79        layout->addLayout(new QVBoxLayout, j, i);
80        auto l = layout->itemAtPosition(j, i)->layout();
81        for (auto k = 1u; k < 6; ++k) {
82          auto w = new QWidget(this);
83          auto vl = new QVBoxLayout(w);
84
85          QLabel *lab;
86
87          auto type = rand() % 4;
88
89          if (type == 3) {
90            lab = new QLabel("");
91            lab->setStyleSheet(style + QString(" padding: 32px;"));
92          } else {
93            lab = new QLabel("Subject");
94            lab->setStyleSheet(style +
95                               QString("font-size: 15px; font-weight:
                                      800; "));
96            lab->setAlignment(Qt::AlignCenter);
97
98            vl->addWidget(lab);
99
100           lab = new QLabel(QString("Type%1").arg(k));
101           lab->setStyleSheet(style + QString("font-size: 12px;"));
102           lab->setAlignment(Qt::AlignCenter);
103           vl->addWidget(lab);
104           lab = new QLabel(temp);
105
106           lab->setStyleSheet(style + QString("font-size: 12px;"));
```

```
107           lab->setAlignment(Qt::AlignCenter);
108           vl->addWidget(lab);
109           lab = new QLabel("Class");
110           lab->setStyleSheet(style + QString("font-size: 12px;"));
111           lab->setAlignment(Qt::AlignCenter);
112         }
113
114         vl->addWidget(lab);
115         vl->setSpacing(0);
116         //        vl->setContentsMargins(15, 15, 15, 15);
117         w->setLayout(vl);
118         auto el =
119             new elem((i == 0 ? k : -1), type, (i == 0 ? delete w,
                nullptr : w));
120
121         el->setProperty("group", QString("G%1").arg(i));
122         el->setProperty("layout", k);
123         el->setProperty("row", j);
124         el->setProperty("col", i);
125         l->addWidget(el);
126
127         connect(el, &elem::rigthButtonClicked, [this](elem *w) {
128           if (toSwap.first == nullptr)
129             toSwap.first = w;
130           else {
131             elemSwap(toSwap.first, w);
132             toSwap.first = nullptr;
133           }
134         });
135
136         connect(el, &elem::leftButtonClicked, [this](elem *w) {
137           if (w != currentSelected) {
138             if (currentSelected != nullptr) {
139               leftClickAnimation(currentSelected, -20, 1);
140             }
141             leftClickAnimation(w, 20, 0);
142             currentSelected = w;
```

```
143             } else {
144               leftClickAnimation(w, -20, 1);
145               currentSelected = nullptr;
146             }
147           });
148         }
149       }
150     }
151
152     content->setLayout(layout);
153     //  layout->setMargin(0);
154     layout->setContentsMargins(0, 1, 1, 1);
155     //  layout->removeItem(layout->itemAtPosition(0, 0));
156     setWidget(content);
157 }
158
159 Schedule::~Schedule() {}
160
161 void elem::mousePressEvent(QMouseEvent *event) {
162     if (event->button() == Qt::MouseButton::RightButton) {
163       emit rigthButtonClicked(this);
164     } else if (event->button() == Qt::MouseButton::LeftButton) {
165       emit leftButtonClicked(this);
166     }
167 }
```

```
1  #ifndef SCHEDULE_H
2  #define SCHEDULE_H
3
4  #include <QColor>
5  #include <QDebug>
6  #include <QGridLayout>
7  #include <QLabel>
8  #include <QPair>
9  #include <QParallelAnimationGroup>
10 #include <QPropertyAnimation>
11 #include <QScrollArea>
12 #include <QWidget>
```

```
13
14 const QColor practice_active("#8043E8");
15 const QColor practice("#F5EAFE");
16
17 const QColor lab_active("#FF5D74");
18 const QColor lab("#FEE8EB");
19
20 const QColor lecture_active("#FFC060");
21 const QColor lecture("#F5EAFE");
22
23 const QColor gap("FFFFFF");
24
25 class elem : public QWidget {
26   Q_OBJECT
27
28   QHBoxLayout* layout;
29   QWidget* content;
30   QString _type = "";
31   QString _class = "";
32   QString _title = "";  // group teacher
33   QString _subj = "";
34
35  public:
36   QString lectureStyle = QString(
37       "border-left: 2px solid #FFC060;"
38       "background-color: #FFF4CF;"
39       "padding: 2px 40px;");
40
41   QString practiceStyle = QString(
42       "border-left: 2px solid #8043E8;"
43       "background-color: #E9D7FA; "
44       "padding: 2px 40px;");
45
46   QString labStyle = QString(
47       "border-left: 2px solid #FF5D74;"
48       "background-color: #FEE8EB; "
49       "padding: 2px 40px;");
```

```
50
51  QString gapStyle = QString(
52      "border:none;"
53      "background-color: #FFFFFF; ");
54
55  QString side = QString(
56      "border:none;"
57      "background-color: white; "
58      "padding: 2px 2px;");
59
60  uint type;
61  elem(int index, uint type, QWidget* w) : content(w), type(type) {
62    layout = new QHBoxLayout(this);
63    if (index != -1) {
64      auto label = new QLabel(QString::number(index));
65      label->setStyleSheet(side);
66      label->setSizePolicy(QSizePolicy::Maximum, QSizePolicy::Minimum
          );
67      layout->insertWidget(layout->count(), label);
68      if (!w) {
69        content = label;
70        content->setProperty("hello", "world");
71      }
72    }
73    if (w) layout->insertWidget(layout->count(), w);
74    layout->setContentsMargins(2, 2, 2, 2);
75
76    if (type == 0) {
77      setStyleSheet(lectureStyle);
78    } else if (type == 1) {
79      setStyleSheet(practiceStyle);
80    } else if (type == 2) {
81      setStyleSheet(labStyle);
82    } else {
83      //      setStyleSheet(gapStyle);
84    }
85    //    if (index == -1) {
```

```
86      //      setStyleSheet(gapStyle);
87      //    }
88
89      setLayout(layout);
90    }
91
92    void removeContent() { layout->removeWidget(content); }
93
94    void setContent(QWidget* w) {
95      layout->insertWidget(layout->count(), w);
96      content = w;
97    }
98
99    QWidget* getContent() { return content; }
100
101  signals:
102    void rigthButtonClicked(elem* w);
103    void leftButtonClicked(elem* w);
104
105  protected:
106    virtual void mousePressEvent(QMouseEvent* event) override;
107 };
108
109 class Schedule : public QScrollArea {
110    Q_OBJECT
111
112    QGridLayout* layout;
113    QWidget* content;
114    elem* currentSelected = nullptr;
115    void leftClickAnimation(elem* w, int value, bool reset);
116    QPair<elem*, elem*> toSwap = {nullptr, nullptr};
117
118    uint num = 0;
119
120  public:
121    bool teacher_schedule_flag;
122    Schedule(QWidget* parent = nullptr);
```

```
123
124   void elemSwap(elem* l, elem* r) {
125     if (l->getContent()->property(”hello”).toByteArray() == ”world”
          ||
126         r->getContent()->property(”hello”).toByteArray() == ”world”)
127       return;
128
129     auto p1 = new QPropertyAnimation(l, ”geometry”);
130     p1->setEasingCurve(QEasingCurve::InOutCubic);
131     p1->setStartValue(
132         QRect(l->pos().x(), l->pos().y(), l->width(), l->height()));
133     p1->setEndValue(QRect(r->pos().x(), r->pos().y(), l->width(), l->
          height()));
134     p1->setDuration(700);
135
136     auto p2 = new QPropertyAnimation(r, ”geometry”);
137     p2->setEasingCurve(QEasingCurve::InOutCubic);
138     p2->setStartValue(
139         QRect(r->pos().x(), r->pos().y(), l->width(), l->height()));
140     p2->setEndValue(QRect(l->pos().x(), l->pos().y(), l->width(), l->
          height()));
141     p2->setDuration(700);
142
143     auto gr = new QParallelAnimationGroup;
144     gr->addAnimation(p1);
145     gr->addAnimation(p2);
146
147     gr->start(QAbstractAnimation::DeleteWhenStopped);
148   }
149   ~Schedule();
150   bool getTeacher_schedule_flag() const;
151   void setTeacher_schedule_flag(bool value);
152   uint getNum() const;
153   void setNum(const uint& value);
154 };
155 #endif  // SCHEDULE_H
```

```
1 #ifndef SIDEBAR_H
```

```
2  #define SIDEBAR_H
3
4  //#include <QtGui/QWidget>
5  #include <QAction>
6  #include <QIcon>
7  #include <QPaintEvent>
8  #include <QPainter>
9
10 class SideBar : public QWidget {
11   Q_OBJECT
12
13  public:
14   QAction *_checkedAction;
15   SideBar(QWidget *parent = 0);
16   ~SideBar();
17
18   void addAction(QAction *action);
19   QAction *addAction(const QString &text, const QIcon &icon = QIcon()
        );
20
21   QList<QAction *> actions();
22
23  protected:
24   void paintEvent(QPaintEvent *event);
25   void mousePressEvent(QMouseEvent *event);
26   void mouseReleaseEvent(QMouseEvent *event);
27   QSize minimumSizeHint() const;
28
29   QAction *actionAt(const QPoint &at);
30
31  private:
32   void animateSlideAndFade(QRect *rect);
33   QList<QAction *> _actions;
34   QAction *_pressedAction;
35 };
36
37 #endif  // SIDEBAR_H
```

```
1  #include ”sidebar.h”
2
3  #include <QGraphicsRectItem>
4  #include <QPropertyAnimation>
5  #include <QVariantAnimation>
6
7  int action_height = 60;
8  const QColor activeBgColor(”#EFEFEF”);
9  const QColor bgColor(”#2A2D34”);
10 const QColor textColor(”#BEDCFE”);
11 const QColor activeTextColor(”#2A2D34”);
12 const QColor activeColor(”#2667FF”);
13
14 SideBar::SideBar(QWidget *parent)
15     : QWidget(parent), _pressedAction(NULL), _checkedAction(NULL) {
16   setFixedWidth(160);
17
18   //  setFixedHeight(800);
19 }
20
21 SideBar::~SideBar() {}
22
23 void SideBar::paintEvent(QPaintEvent *event) {
24   QPainter p(this);
25   QFont fontText(p.font());
26   fontText.setFamily(”Noto Sans CJK”);
27   fontText.setPixelSize(14);
28   p.setFont(fontText);
29
30   int actions_height = _actions.size() * action_height;
31
32   int action_y = event->rect().height() / 2 - actions_height / 2;
33   for (int i = 0; i < _actions.size(); ++i) {
34     QRect actionRect(0, action_y, event->rect().width(),
          action_height);
35
36     if (_actions[i]->isChecked()) {
```

```
37        QRect *activePlug = new QRect(4, action_y + 4, 6, action_height
              - 7);
38        p.fillRect(actionRect, activeBgColor);
39        p.fillRect(*activePlug, activeColor);
40      }
41
42      if (!_actions[i]->isChecked()) {
43        p.fillRect(actionRect, bgColor);
44      }
45
46      // icon
47      int icon_size = 80;
48
49      QRect actionIconRect(0, action_y, event->rect().width(),
50                           action_height - 20);
51      QIcon actionIcon(_actions[i]->icon());
52      actionIcon.paint(&p, actionIconRect);
53
54      p.setPen(QColor(217, 217, 217));
55      if (_actions[i]->isChecked()) {
56        p.setPen(bgColor);
57      }
58      QRect actionTextRect(35, action_y + actionRect.height() - 37,
59                           event->rect().width(), 15);
60      p.drawText(actionTextRect, Qt::AlignVCenter, _actions[i]->text())
           ;
61
62      action_y += actionRect.height();
63    }
64  }
65
66  QSize SideBar::minimumSizeHint() const {
67    return QSize(90, _actions.size() * action_height);
68  }
69
70  void SideBar::addAction(QAction *action) {
71    _actions.push_back(action);
```

```
72    action->setCheckable(true);
73    update();
74    return;
75  }
76
77  QAction *SideBar::addAction(const QString &text, const QIcon &icon) {
78    QAction *action = new QAction(icon, text, this);
79    action->setCheckable(true);
80    _actions.push_back(action);
81    update();
82    return action;
83  }
84
85  void SideBar::mousePressEvent(QMouseEvent *event) {
86    _pressedAction = actionAt(event->pos());
87    if (_pressedAction == NULL || _pressedAction == _checkedAction)
        return;
88    update();
89  }
90
91  void SideBar::mouseReleaseEvent(QMouseEvent *event) {
92    QAction *tempAction = actionAt(event->pos());
93    if (_pressedAction != tempAction || tempAction == NULL) {
94      _pressedAction = NULL;
95      return;
96    }
97    if (_checkedAction != NULL) _checkedAction->setChecked(false);
98    _checkedAction = _pressedAction;
99    if (_checkedAction != NULL) _checkedAction->setChecked(true);
100   update();
101   _pressedAction = NULL;
102   return;
103 }
104
105 QAction *SideBar::actionAt(const QPoint &at) {
106   int actions_height = _actions.size() * action_height;
107
```

```
108   int action_y = rect().height() / 2 - actions_height / 2;
109   foreach (QAction *action, _actions) {
110     QRect actionRect(0, action_y, rect().width(), action_height);
111     if (actionRect.contains(at)) return action;
112     action_y += actionRect.height();
113   }
114   return NULL;
115 }
116
117 void SideBar::animateSlideAndFade(QRect *rect) {
118   QPropertyAnimation *a1 =
119       new QPropertyAnimation(((QWidget *)rect), "geometry");
120
121   a1->setDuration(200);
122   auto pos = ((QWidget *)rect)->pos();
123   auto size = rect->size();
124   a1->setStartValue(QRect(pos.x(), pos.y() + 100, size.width(), size.
        height()));
125   a1->setEndValue(QRect(pos.x(), pos.y(), size.width(), size.height()
        ));
126
127   a1->start(QPropertyAnimation::DeleteWhenStopped);
128 }
129
130 QList<QAction *> SideBar::actions() { return _actions; }
```