Міністерство освіти і науки України

Національний університет «Києво-Могилянська академія»

Факультет інформатики

Кафедра інформатики

Курсова робота

освітній ступінь – бакалавр

на тему: «**ВИКОРИСТАННЯ МЕТА-ЕВРИСТИЧНИХ АЛГОРИТМІВ ДЛЯ ВИРІШЕННЯ ЗАДАЧ НА ЗАДОВОЛЕННЯ ОБМЕЖЕНЬ**»

Виконав: студент 4-го року навчання,

Спеціальності
122 Комп’ютерні науки

Орел Даниїл Миколайович

Керівник Бабич Т.А., _____
магістр комп’ютерних наук, асистент
«____» _____ 20____ р.

Київ – 2022

Національний університет «Києво-Могилянська академія»

Факультет інформатики

Кафедра інформатики

Освітній ступінь бакалавр

Спеціальність 122 Комп’ютерні науки

Освітня програма бакалавр

ЗАТВЕРДЖУЮ

Завідувач кафедри інформатики

Гороховський С. С.

“____” _____20___року

ЗАВДАННЯ

ДЛЯ КУРСОВОЇ РОБОТИ СТУДЕНТУ

Орлу Даниїлу Миколайовичу

1. Тема роботи Використання мета-евристичних алгоритмів для вирішення задач на задоволення обмежень, керівник роботи Бабич Трохим Анатолійович, магістр комп’ютерних наук, асистент

2. Строк подання студентом роботи 20.05.2022

3. Анотація

Вступ

Розділ 1. Дослідження та аналіз предметної області

1.1. Опис предметної області та збір фактів для повноти моделі

1.2. Визначення факторів, що визначають ефективність моделі

1.3. Математичний опис проблеми задачі задоволення обмежень

Розділ 2. Проектування та розробка сервісу планування

2.1. Пакет OptaPlanner для вирішення задачі задоволення обмежень

ГРАФІК ПІДГОТОВКИ КУРСОВОЇ РОБОТИ ДО ЗАХИСТУ

№ з/п	ПЕРЕЛІК РОБІТ	Термін виконання	Дата ознайомлення наукового керівника	Підпис наукового керівника	Примітки
1.	Вибір теми, затвердження її на засіданні кафедри та закріплення наукового керівника Узгодження календарного графіка підготовки кваліфікаційної роботи. Ознайомлення студента з критеріями оцінювання кваліфікаційної роботи (п. 8.5).	21 жовтня			
2.	Вивчення джерел літератури, матеріалів архівів, періодичних видань, збір та узагальнення фактів, даних	30 жовтня – 14 листопада			
3.	Складання плану каліф. роботи та узгодження з науковим керівником	16 листопада			
4.	Написання розділів роботи *або* Постановка експерименту, аналіз отриманих результатів наукового дослідження	20 листопада – 10 квітня			
5.	Проміжний контроль виконання роботи	12 лютого			
6.	Написання кваліфікаційної роботи в цілому, ознайомлення з її першим варіантом наукового керівника	9 березня – 6 травня			
	Розділ 1 (постановка проблеми, теоретичні основи, огляд літературних джерел)	13 квітня			
	Розділ 2 (аналітично-дослідницька частина) (експериментальна частина для природничих і біологічних наук)	29 квітня			
	Розділ 3 (проектно-рекомендаційна частина) (аналіз результатів експерименту для природничих і біологічних наук)	4 травня			
7.	Повне завершення написання кваліфікаційної роботи, оформлення її згідно з вимогами й подання на відгук науковому керівнику	13 квітня – 20 травня			
8.	Подання кваліфікаційної роботи для перевірки письмових робіт студентів НаУКМА на відповідність вимогам академічної доброчесності,	20 травня			

Графік узгоджено «21» жовтня 2021р.

Науковий керівник Бабич Трохим Анатолійович

Виконавець курсової роботи Орел Даниїл Миколайович

Зміст

Анотація

Дана робота присвячена вирішенню задачі на задоволення обмежень, що слугує фундаментом для побудови ефективної моделі для автогенерації розкладу. Вирішення цієї проблеми дозволить методистам будь-якого факультету в університеті “Києво-Могилянської академії” отримувати готовий навчальний розклад.

У роботі представлено модель, що вирішує задачу задоволення обмежень, побудовану на основі інструментарію OptaPlanner. Окрім моделі, було висвітлено нову архітектуру веб-конструктора для побудови навчального розкладу, яка включає у себе реалізацію сервісу для автогенерації розкладу.

У висновках було проведено аналіз ефективності побудованої моделі та подальші шляхи її вдосконалення.

Вступ

Кожного дня, люди зіштовхуються з проблемами обмежених ресурсів для вирішення задач як особистісного, локального рівня (для прикладу, певних буденних задач, що потребують якісного тайм менеджменту), так і глобального рівня, вирішуючи оптимізації корпоративних ресурсів (гроші, активи, працівники, і звичайно час). Сукупність такого роду задач утворюють фундаментальну математичну проблему, яку називають задачею на задоволення обмежень, яка слугує для максимізації ефективності використання певних ресурсів.

У даній роботі було розглянуто один з прикладів такого роду задач, а саме планування розкладу на певний навчальний семестр. Як майданчик для вирішення такої задачі та постановки експериментів було обрано Національний Університет Києво-Могилянської Академії (далі НаУКМА).

Плануючи навчальний розклад, методисти зіштовхуються з багатьма проблемами, оскільки існує багато обмежень, які потрібно врахувати при побудові розкладу, а саме: накладання пар, накладання місць проведення занять, преференції або ж вимоги викладачів щодо часу та місця проведення пар, тощо.

Мануальна конфігурація розкладу не дає можливості математично та детерміністично оцінити наскільки якісно побудована конфігурація. Окрім уникнення мануальної частини по складанню розкладу, автоматизоване рішення дозволить методистам аналізувати ефективність згенерованого плану по навчальному семестру, та порівнювати з іншими, можливими його конфігураціями.

Розділ 1. Дослідження та аналіз предметної області

1.1. Опис предметної області та збір фактів для повноти моделі

Планування навчального періоду є комплексною математичною задачею. Вирішення цієї проблеми дозволить закріпити за побудованою навчальною програмою певний часовий проміжок, протягом якого студенти мають здобути знання з визначених дисциплін.

Кожен процес використовує ресурси, які дозволяють вирішити поставлені цілі. Навчання це в першу чергу процес, а отже воно використовує ресурси, які залучені в цей процес, а саме:

- Людські
- Часові
- Фінансові

Для побудови навчального розкладу, використовуються людські та часові ресурси. Визначимо основних учасників навчального процесу, їхні ролі та вимоги:

- Адміністрація університету
    1. Складають навчальну програму та визначають набір предметів, які будуть викладатись протягом навчального періоду;
    2. Займаються пошуком кваліфікованих викладачів, які будуть викладати визначені дисципліни;
    3. Відповідають за стан та готовність кабінетів до навчального процесу;
    4. Відповідають за стан та наявність інструментів, потрібних для освоєння дисциплін.
- Викладачі

1. Маючи певну кваліфікацію, виконують план навчальної програми по певному предмету за зручним для них графіком роботи;
2. Використовують надані університетом умови та інструменти для проведення занять. Якщо предмет специфічний, викладачі мають преференсії та пріорітетність щодо місця проведення пар, оскільки не кожен кабінет може бути оснащений потрібними інструментами.

- Студенти
    1. Здобувають знання за планом навчальної програми по певних предметах за визначеним для них графіком, який не повинен накладатись з іншими предметами.

Пандемія Covid-19 внесла певні корективи до навчального процесу [1] в усіх навчальних закладах по всіх куточках світу, зокрема і в НаУКМА. Більшість студентів (студенти 3-го та 4-го бакалаврської програми, та студенти 2-го курсу програми магістрів) навчаються дистанційно, решту студентів навчаються за змішаною системою навчання.

Нові умови сприяють новим підходам та вимогам, проте в даній роботі буде розглянуто формат офлайн навчання. Він є більш складним в порівнянні з форматом дистанційного навчання, оскільки у ньому буде враховано вимогу викладачів щодо аудиторії, в якій буде проводитись навчання студентів. У моделі дистанційного навчання це обмеження відсутнє, та викладачі самостійно обирають зручне робоче місце, і проводять лекції та практики в онлайн платформах, як Microsoft Teams, Zoom, Google Meets, тощо.

1.2. Визначення факторів, що визначають ефективність моделі

Кожна проблема планування має оптимальні цілі, які базуються на лімітованості ресурсів та пріорітетності вимог учасників процесу. Оптимальними цілями може бути будь-яка кількість факторів, за якими оцінюється ефективність моделі [2]. Базуючись на отриманих фактах з предметної області (планування навчального розкладу в НаУКМА) можна виявити найбільш впливові фактори:

- Мінімізація збігів дисциплін (далі MSI) – студенти, як і викладач *не повинні* мати дві або більше дисципліни, що проходять в один і той самий час;
- Максимізація задоволення викладачів (далі MPS) – викладачам з певними преференсіями щодо часу та місця викладання *слід* надати пріорітет. Викладачі також *полюбляють* мати пари без вікон.

При дослідженні формату офлайн навчання, для обрахунку ефективності побудови розкладу необхідно включити фактори, що стосуються локації, тобто аудиторій, в яких проводитимуться заняття:

- Мінімізація збігів аудиторій (далі MAI) – аудиторії *не повинні* бути зайняті двома і більше дисциплінами одночасно;
- Максимізація практичності (далі MP) – викладачам з специфічними дисциплінами, які потребують певного спорядження, *повинні* надати кабінети з пріорітетом;

Зазначені фактори й трактуються обмеженнями [2], і вони поділяються на декілька категорій:

- Жорстке обмеження (hard constraint) не повинно бути порушено ні в якому разі. Прикладами є фактори MSI, MAI та MP.
- М’яке обмеження (soft constraint) не слід порушувати, якщо це можна уникнути. Прикладом слугує коефіцієнт MPS.

Окрім жорсткості обмежень, існує поділ на:

- Позитивність обмеження (positive constraint) – варто виконати, якщо це можливо. Прикладом слугує м’який коефіцієнт MPS, оскільки викладам *слід* надати преференсії по можливості.
- Негативність обмеження (negative constraint) – не варто виконувати, якщо це можливо. Як приклад, візьмемо жорсткий коефіцієнт MAI, що свідчить про *неможливість* збігу аудиторій.

Сукупність обмежень визначає оцінку ефективності моделі. Можна дійти до висновку, що чим більше було задоволено обмежень, тим краще рішення було знайдене.

Варто зазначити, що ефективність моделі, відповідно й рівень задоволеності обмежень буде залежати від кількості доступних ресурсів у вибірці. Аналізуючи предметну область, можна підсумувати від чого залежать результати моделі:

- Кількості викладачів та вимог від них: чим більше викладачів, які мають певні преференсії, тим коефіцієнт MPS буде меншим, оскільки не кожен викладач зможе отримати бажану вимогу через м’якість обмеження;

1.3. Математичний опис проблеми задачі задоволення обмежень

В основі задачі побудови навчального розкладу лежить математична проблема – задача виконання обмежень [6].

Нехай існує певна сукупність об’єктів, стан яких має задовольняти певні обмеження. Формально, цю задачу можна представити у вигляді трійки <X, D, C>, де X - множинна змінних, D - область значень, а C - множина обмежень. Кожне обмеження у свою чергу є парою <t, R>, де t - кортеж, що складається з n змінних, а R - n-місне відношення D.

Оцінка змінної - функція, що відображає множину змінних на область значень D. Це можна записати у такій формі: f: X → D.

Розв’язком цієї задачі буде оцінка, що задовольняє всім обмеженням, що присутні у сукупності об'єктів. Існує така класифікація можливих розв’язків цієї задачі [2]:

- Можливим рішенням (possible solution) є будь-яке рішення, незалежно від того, порушує воно будь-яку кількість обмежень чи ні. Проблеми планування мають надзвичайно велику кількість можливих рішень. Багато з цих рішень є неефективними.
- Доступне рішення (feasible solution) – це рішення, яке не порушує жодних (негативних) жорстких обмежень. Іноді немає доступних рішень. Кожне доступне рішення є можливим рішенням.
- Оптимальним рішенням (optimal solution) є рішення з найвищим балом ефективності. Задачі з плануванням мають 1 або декілька оптимальних рішень. Завжди існує принаймні 1 оптимальне рішення, навіть якщо доступних рішень немає, а оптимальне рішення не є доступним рішенням.
- Найкраще рішення (best solution) – це рішення з найвищим балом ефективності, знайденим реалізацією за певний проміжок часу. Найкраще знайдене рішення буде доступим, і, за умови поліномільності часу виконання алгоритму, це є оптимальним рішенням.

Для того, щоб знайти оптимальне, або ж найкраще рішення розкладу, необхідно перевірити та оцінити усі можливі конфігурації аудиторій, викладачів, часу початку та кінця пари, й інших параметрів на сумісність з наявними в моделі обмеженнями.

Перший інтуїтивний варіант, яких спадає на думку – зробити повний перебір усіх можливих рішень, та обрати з них оптимальне рішення, проте кількість можливих рішень навіть на невеликому наборі даних може сягати за кількість атомів у видимому Всесвіті. Це пов’язано з тим, що при додаванні нового кортежу обмежень <t, R>, або ж при зростанні множини змінних X, кількість можливих рішень експоненційно зростає.

Можемо дійти до висновку, що проблема планування навчального розкладу є NP-повною, що означає [2, 3]:

- Легко перевірити знайдене рішення задачі за поліномільний час.
- Проте немає алгоритму, щоб знайти найкраще рішення проблеми за поліноміальний час.

Розділ 2. Проектування та розробка сервісу планування

2.1. Пакет OptaPlanner для вирішення задачі задоволення обмежень

Повний перебір усіх можливих рішень для пошуку найкращого рішення, є коректним, проте алгоритмічно невирішуваним процесом [3]. Потрібен інструмент, який за допомогою оптимізацій фільтрує підмножину можливих неефективних рішень, і опрацьовує якомога якісніші, доступні рішення, які можуть стати оптимальними.

Одним з таких інструментів є OptaPlanner, який ефективно пробиратися через цю неймовірно велику кількість можливих рішень за допомогою кілька прошарків оптимізацій та евристик [2, 4], таких як: Tabu Search, Simulated Annealing, Late Acceptance, та інші алгоритми зі сфері штучного інттелекту. Залежно від набору даних та обмежень, деякі алгоритми оптимізації працюють краще, ніж інші, але це неможливо спрогнозувати заздалегідь.

OptaPlanner підтримує такі можливості [4]:

- Постійне планування (continious planning) для щотижневої публікації розкладу;
- Безперебійне перепланування (non-distruptive planning) змін до вже опублікованого розкладу;
- Планування в режимі реального часу (real-time planning) для реагування на збої в плані в режимі реального часу;
- Надмірне планування (overconstrained planning), коли ресурсів занадто мало;
- Корегованість (pinning), щоб користувач все ще міг вносити зміни вручну у розклад.

2.2. Моделювання та реалізація сервісу планування розкладу з OptaPlanner

Предметна область планування навчального розкладу виокреслює чотири найголовніших сутності, навколо яких вибудовується уся логіка автогенерації:

- Предмет (Lesson)
- Аудиторія (Room)
- Пара (Timesplot)
- Розклад (TimeTable)

Побудуємо діаграму взаємозв’язків між зазначеними сутностями (див. рис. 1).

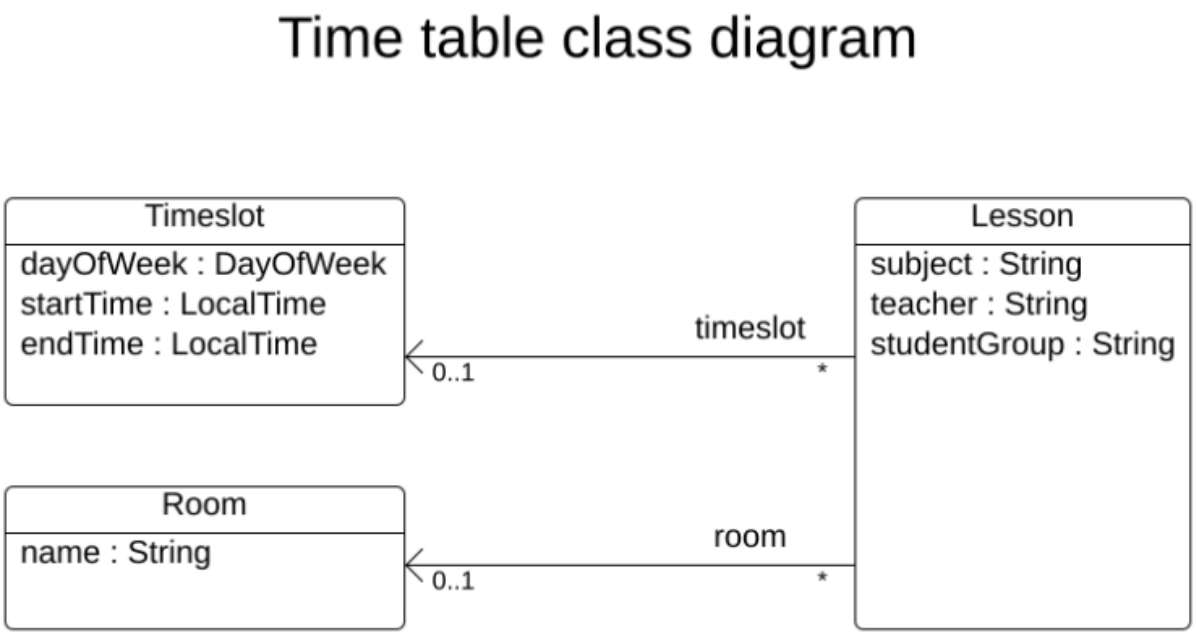


Рис. 1: Діаграма взаємозв’язків між сутностями навчального розкладу

Як можна побачити, зв’язок між Предметом, Аудиторією та Парою відноситья як один до багатьох, оскільки:

- У одній аудиторії може проводитись жоден, один або багато предметів;

- У одній парі може проводитись багато предметів.

Але виникає проблема, коли у одній аудиторії, в один той самий часовий слот може проводитись одночасно декілька предметів.

Для вирішення такого роду конфліктів застосуємо ряд м’яких та жорстких обмежень з предметної області. З м’яких обмежень можна виділити:

- MTS (Максимізація сталості викладання) є позитивним м’яким обмеженням (див. додаток А);
- MTW (Мінімізація вікон викладання) є позитивним м’яким обмеженням (див. додаток Б);
- MSV (Максимізація різноманіття предметів) є негативним м’яким обмеженням (див. додаток В).

Для повноти опису жорстких обмежень з предметної області, потрібно подрібнити вже виявлені обмеження на більш детальні, щоб отримати якомога якіснішу модель:

- MTC (Мінімізація конфліктів викладання) є негативним жорстким обмеженням (див. додаток Г);
- MAC (Мінімізація конфліктів аудиторій) є негативним жорстким обмеженням (див. додаток Д);
- MSC (Мінімізація конфліктів відвідування) є негативним жорстким обмеженням (див. додаток Е).

За допомогою визначених обмежень, можна сформувати об’єкт Contsraint (див. рис. 2) з бібліотеки OptaPlanner, який в подальшому використовується для оцінювання ефективності побудованої моделі [2].

```
@Override
    public Constraint[] defineConstraints(ConstraintFactory constraintFactory) {
        return new Constraint[] {
                // Hard constraints
                roomConflict(constraintFactory),
                teacherConflict(constraintFactory),
                studentGroupConflict(constraintFactory),
                // Soft constraints
                teacherRoomStability(constraintFactory),
                teacherTimeEfficiency(constraintFactory),
                studentGroupSubjectVariety(constraintFactory)
        };
    }
```

Рис. 2: Набір жорстких та м’яких обмежень для побудови моделі

Модель складається з класу ScoreManager для оцінки ефективності обмежень HardSoftScore (див. рис. 3) в поєднанні з класом SolverManager (див. рис. 4), що має мета-евристики для пошуку кращого розкладу серед можливих.

```
@Inject
ScoreManager<TimeTable, HardSoftScore> scoreManager;
```

Рис. 3: Клас призначений для оцінки ефективності моделі

```
@Inject
SolverManager<TimeTable, Long> solverManager;
```

Рис. 4: Клас призначений для пошуку ефективних рішень

Використовуючи вище зазначений інструментарій пакету OptaPlanner було побудовано функціонал для планування розкладу, використовуючи основні сутності з предметної області.

Перед реалізацією сервісу було передбачено такого роду функціональності [2]:

- Табличне формування розкладу з можливістю представлення даних навколо різних сутностей (див. додаток Є);
- Мануальне внесення та редагування даних (див. додаток Ж, З та И);
- Зупинка планування розкладу на найкращому рішенні з усіх знайдених;
- Очищення дашборду планування розкладу від даних.

2.3. Аналіз результатів побудованої моделі

Розглянемо різні випадки результативності побудованого сервісу базуючись на різноманітті вхідних даних.

Позначимо кількість реляцій Предметів за $|S|$ з області значень X. Добуток множин Аудиторій (A) та Пар (T) утворюють множину усіх можливих перестановок предметів – множину D.

Побудуємо графік у трьохвимірному просторі (див. рис. 5), у якому параметри моделі залежать один від одного.

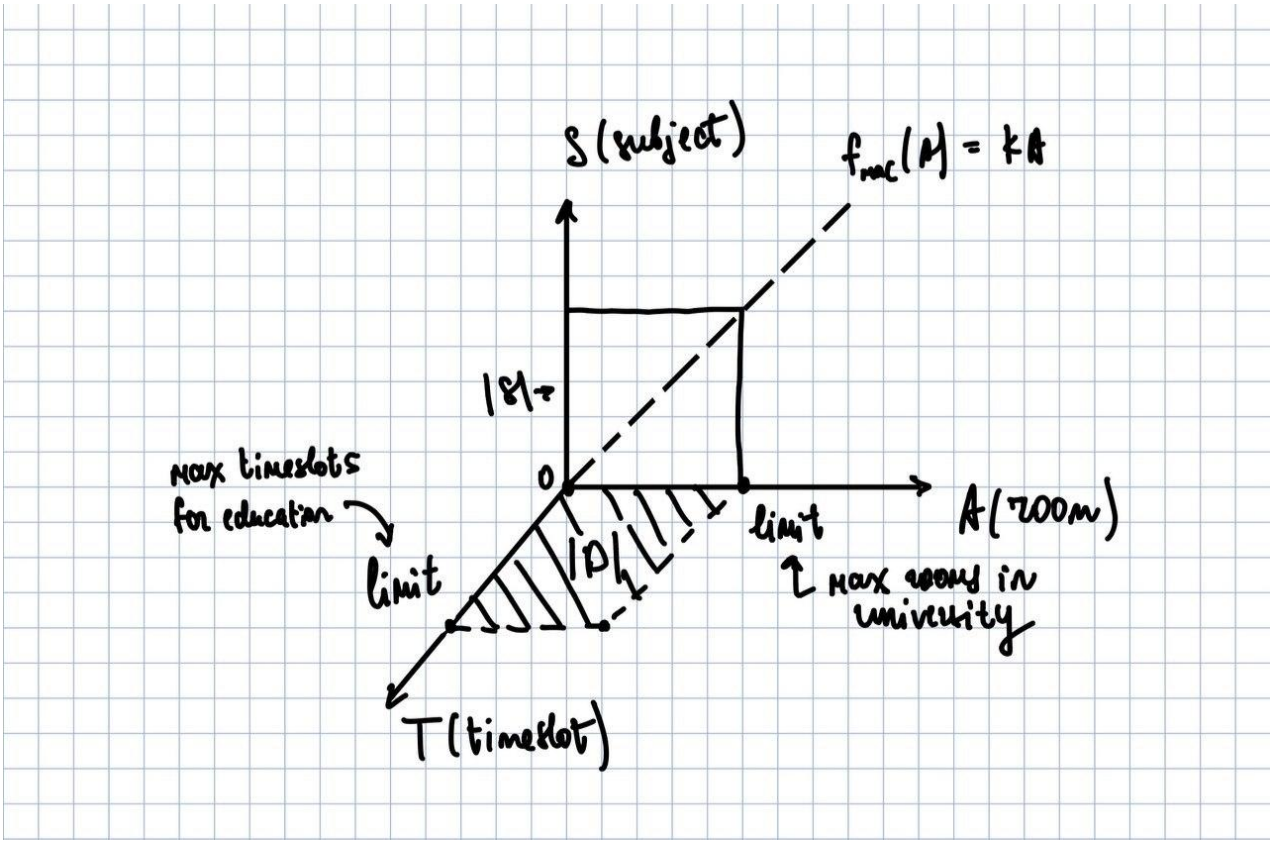


Рис. 5: Графік залежностей між сутностями

Оскільки кількість реляцій пар є строго лімітована (не можна збільшити кількість днів у тижні та час проведення пар не може бути більшим за 24 години в день), відповідно $|T|$ – є обмеженою величиною. Аналогічних властивостей набуває величина $|A|$, оскільки в університеті фіксована кількість аудиторій (за винятком, коли є можливість добудувати нові будівлі та кабінети). Отже величина $|D|$ наближається до певного значення, і не може збільшуватись – вона залежить від кожного університету індивідуально.

Проведемо аналіз однієї з жорстких обмежень MAC (Мінімізація конфліктів аудиторій) між величинами $|S|$ та $|D|$ відповідно. Нехай $|S| = |D|$, а отже й $|S| = |T| * |A|$. Оскільки величина $|T|$ набуває обмеженого значення,

при збільшенні $|S|$ потрібно збільшити й $|A|$, для того щоб конфліктів аудиторій не було. Настає момент, коли $|A|$ також набуває лімітованого значення, й відповідно, $|S|$ набуває більшого значення ніж $|D|$.

Для прикладу, візьмемо конфігурацію з мінімальним набором сутностей з предметної області (див. додаток Є). У цьому випадку, коли $|S| <= |D|$, буде знайдено доступне, а отже й найкраще рішення (див. додаток І). Оцінка цього прикладу складає 0 hard/4soft, що означає, що знайдено оптимальне рішення.

У випадку, коли $|S| > |D|$, $|D| = |T| * |A|$, це означає, що модель з обмеженням MAC не має доступних значень, відповідно не існує оптимального та найкращого рішення, бо порушено жорстку умову MAC (див. додаток Ї). Оцінка такого прикладу складає -1hard/3soft.

Отже, оцінка моделі залежить від кількості жорстких обмежень та від розміру вхідних даних [2]. Оцінка ефективності є більшою у тому разі, коли було використано усі можливі сутності, не було порушено умов жорстких обмежень та мінімізовано порушення м’яких обмежень.

2.4. Інтеграція сервісу планування в екосистему застосунку розкладу

Базуючись на попередній курсовій роботі, було допрацьовано нову схему комунікації та взаємодії між сервісами, які беруть участь у екосистемі застосунку розкладу НаУКМА (див. рис. 6).

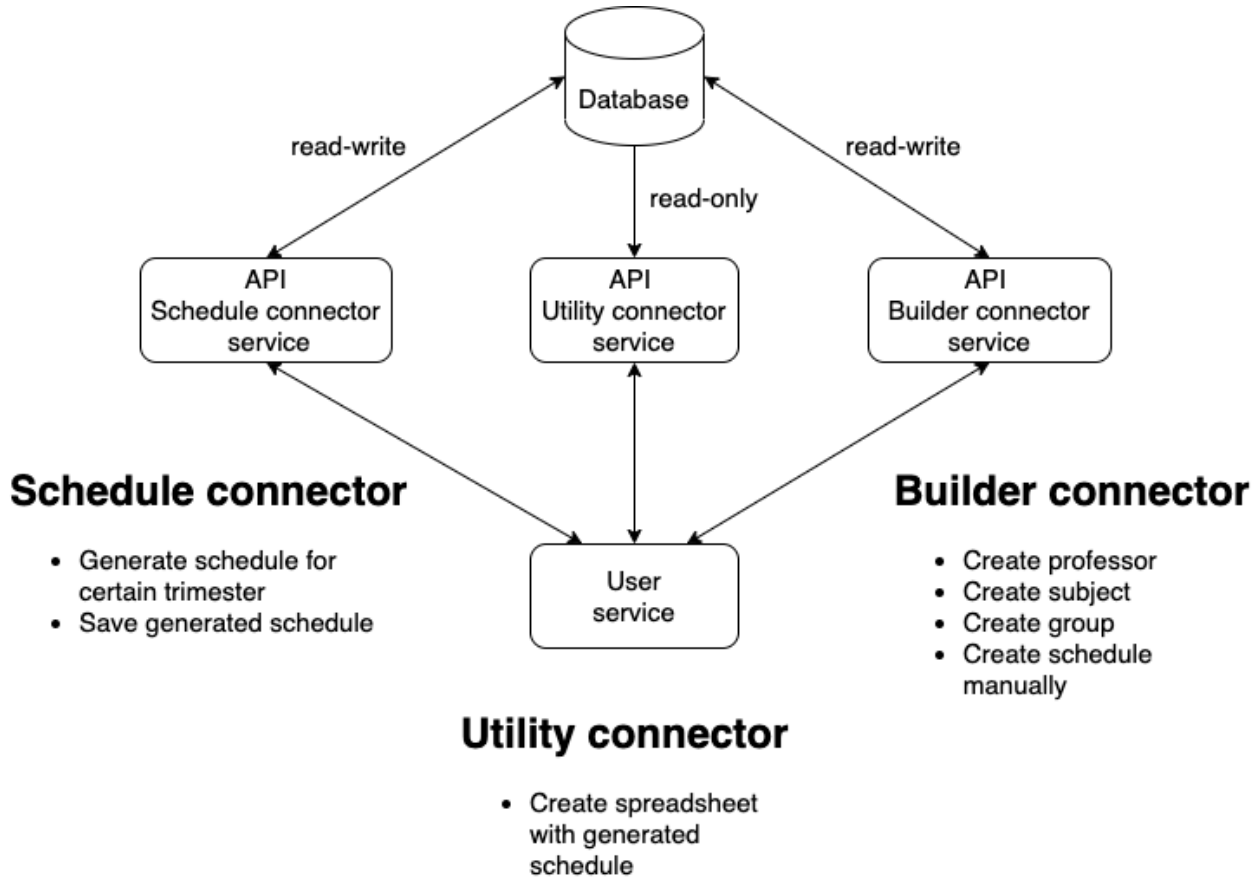


Рис. 6: Архітектура екосистеми розкладу НаУКМА

З’являється новий сервіс, API Schedule connector service, який взаємодіє з PostgreSQL базою даних на правах читання та писання [5]. Таке налаштування ролей дозволяє вибирати необхідні з бази даних сутності Timeslot, Room та Lesson, і базуючись на них – тренувати модель. Окрім цього, така конфігурація дозволяє записувати проміжні результати у сховище, щоб була можливість обирати з оптимальних рішень найкраще для університету.

Опираючись на переваги використання пакету OptaPlanner, є можливість зробити оновлення генерованого розкладу в режимі реального часу. Таким чином, при додаванні нової сутності через інтерфейс, Builder connector service

буде зберігати нові екземпляри сутностей у базі, а це в свою чергу сприятеме виклику нового циклу генерації розкладу.

Висновки

Результатом цієї роботи було проведення побудови сервісу планування навчального розкладу, що вирішує задачу задоволення обмежень. Було проведено математичний аналіз оцінки можливих, доступних, оптимальних та найкращих рішень моделі. Було демонстровано роботу сервісу як самостійне рішення, а також розглянуто подальшу участь застосунку в екосистемі конструктора розкладу.

У подальшому можливі деякі вектори для вдосконалення застосунку, а саме:

- Інтеграція сервісів автогенерації розкладу та конструктора;
- Доповнення та розширення параметрів моделі та обмежень, які допоможуть вирішити задачу одночасно для офлайн, дистанційного та змішаного типів навчання;
- Інтеграція сервісів автогенерації розкладу у екосистему НаУКМА.

Список використаних джерел

1. Preeti Tarkar – Impact of Covid-19 Pandemic On Education System [Електронний ресурс] / Preeti Tarkar – 2020 – Режим доступу до ресурсу: https://www.researchgate.net/profile/Preeti-Tarkar/publication/352647439_Impact_Of_Covid-19_Pandemic_On_Education_System/links/60d1e909299bf19b8d99d279/Impact-Of-Covid-19-Pandemic-On-Education-System.pdf
2. Geoffrey De Smet. – OptaPlanner User Guide [Електронний ресурс] / Geoffrey De Smet – 2022 – Режим доступу до ресурсу: https://docs.optaplanner.org/8.20.0.Final/optaplanner-docs/pdf/optaplanner-docs.pdf
3. Daniel Pierre Bovet, Pierluigi Crescenzy – Introduction to the theory of complexity [Електронний ресурс] – / Daniel Pierre Bover, Pierluigi Crescenzy – 2006 – Режим доступу до ресурсу: https://www.pilucrescenzi.it/wp/wp-content/uploads/2017/09/itc.pdf
4. Geoffrey De Smet. – OptaPlanner User Guide [Електронний ресурс] / Geoffrey De Smet – 2022 – Режим доступу до ресурсу: https://optaplanner.org
5. Yaser Raja – Managing PostreSQL users and roles [Електронний ресурс] / Yaser Raja – 2019 – Режим доступу до ресурсу: https://aws.amazon.com/blogs/database/managing-postgresql-users-and-roles/
6. Rina Dechter – Learning while searching constraint satisfaction problems [Електронний ресурс] / Rina Dechter – 1986 – Режим доступу до ресурсу: https://www.aaai.org/Papers/AAAI/1986/AAAI86-029.pdf?spm=5176.100239.blogcont215226.26.uwOrvt&file=AAAI86-029.pdf&source=post_page

Додатки

Додаток А

М’яке позитивне обмеження: викладач має бажання проводити усі пари в одній аудиторії

```
// A teacher prefers to teach in a single room.
Constraint teacherRoomStability(ConstraintFactory constraintFactory) {
        return constraintFactory
                .forEachUniquePair(Lesson.class,
                        Joiners.equal(Lesson::getTeacher))
                .filter((lesson1, lesson2) -> lesson1.getRoom() != lesson2.getRoom())
                .penalize("Teacher room stability", HardSoftScore.ONE_SOFT);
    }
```

Додаток Б

М’яке позитивне обмеження: викдажач бажає проводити заняття без вікон між його парами

```
// A teacher prefers to teach sequential lessons and dislikes gaps between lessons.
Constraint teacherTimeEfficiency(ConstraintFactory constraintFactory) {
        return constraintFactory
                .forEach(Lesson.class)
                .join(Lesson.class, Joiners.equal(Lesson::getTeacher),
                        Joiners.equal((lesson) -> lesson.getTimeslot().getDayOfWeek()))
                .filter((lesson1, lesson2) -> {
                    Duration between = Duration.between(lesson1.getTimeslot().getEndTime(),
                            lesson2.getTimeslot().getStartTime());
                    return !between.isNegative() && between.compareTo(Duration.ofMinutes(30)) <= 0;
                })
                .reward("Teacher time efficiency", HardSoftScore.ONE_SOFT);
    }
```

Додаток В

М’яке негативне обмеження: студент бажає мати різні пари, що йдуть один за одним

```
// A student group dislikes sequential lessons on the same subject.
Constraint studentGroupSubjectVariety(ConstraintFactory constraintFactory) {
        return constraintFactory
                .forEach(Lesson.class)
                .join(Lesson.class,
                        Joiners.equal(Lesson::getSubject),
                        Joiners.equal(Lesson::getStudentGroup),
                        Joiners.equal((lesson) -> lesson.getTimeslot().getDayOfWeek()))
                .filter((lesson1, lesson2) -> {
                    Duration between = Duration.between(lesson1.getTimeslot().getEndTime(),
                            lesson2.getTimeslot().getStartTime());
                    return !between.isNegative() && between.compareTo(Duration.ofMinutes(30)) <= 0;
                })
                .penalize("Student group subject variety", HardSoftScore.ONE_SOFT);
    }
```

Додаток Г

Жорстке негативне обмеження: викладач мусить проводити пари лише в один час

```
// A teacher can teach at most one lesson at the same time.
Constraint teacherConflict(ConstraintFactory constraintFactory) {
        return constraintFactory
                .forEachUniquePair(Lesson.class,
                        Joiners.equal(Lesson::getTimeslot),
                        Joiners.equal(Lesson::getTeacher))
                .penalize("Teacher conflict", HardSoftScore.ONE_HARD);
    }
```

Додаток Д

Жорстке негативне обмеження: аудиторія повинна бути зайнята лише одним предметом

```
// A room can accommodate at most one lesson at the same time.
Constraint roomConflict(ConstraintFactory constraintFactory) {
        return constraintFactory
                // Select each pair of 2 different lessons ...
                .forEachUniquePair(Lesson.class,
                        // ... in the same timeslot ...
                        Joiners.equal(Lesson::getTimeslot),
                        // ... in the same room ...
                        Joiners.equal(Lesson::getRoom))
                // ... and penalize each pair with a hard weight.
                .penalize("Room conflict", HardSoftScore.ONE_HARD);
    }
```

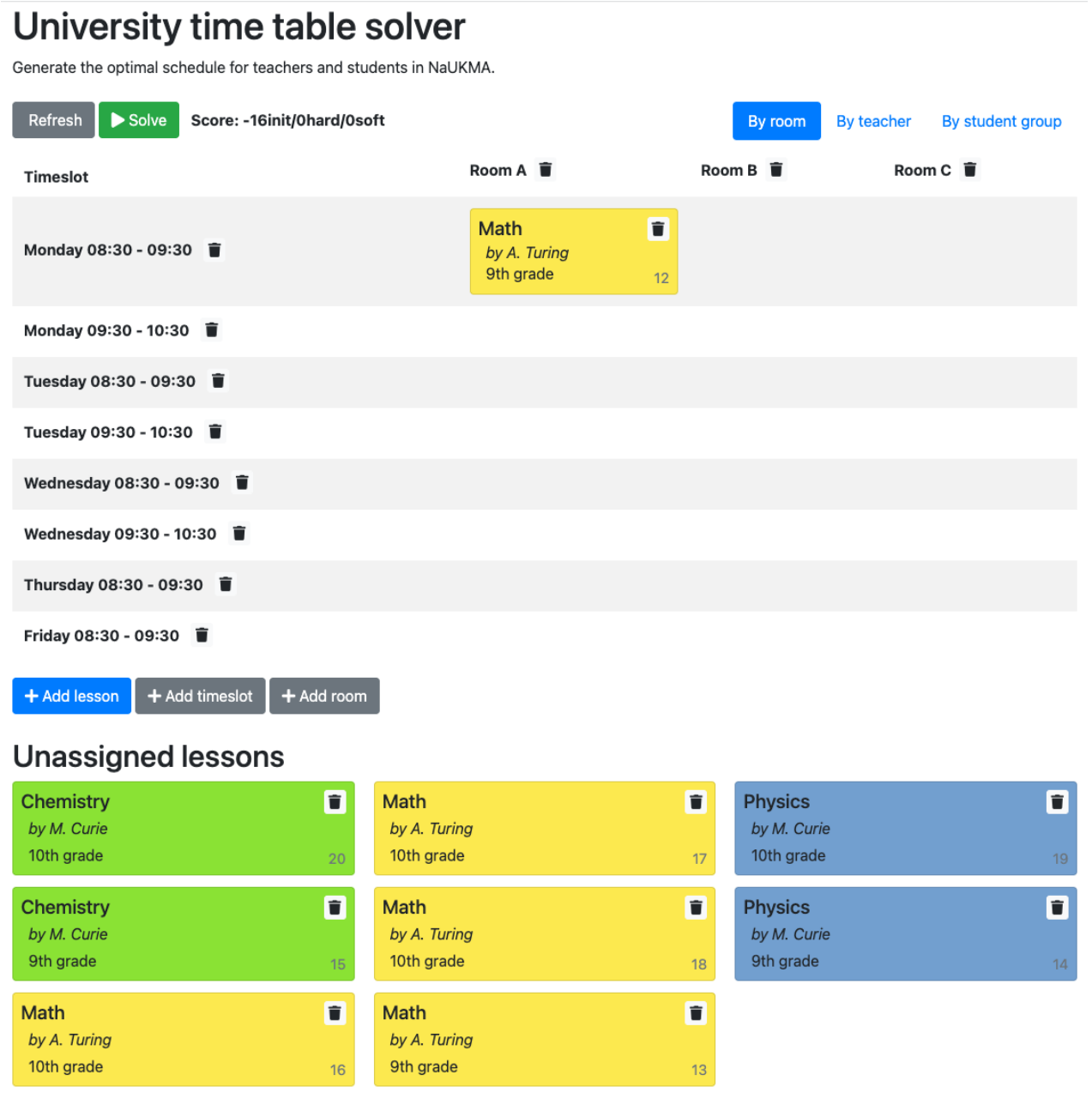
Додаток Е

Жорстке негативне обмеження: студент повинен відвідувати лише одне заняття

```
// A student can attend at most one lesson at the same time.
Constraint studentGroupConflict(ConstraintFactory constraintFactory) {
        return constraintFactory
                .forEachUniquePair(Lesson.class,
                        Joiners.equal(Lesson::getTimeslot),
                        Joiners.equal(Lesson::getStudentGroup))
                .penalize("Student group conflict", HardSoftScore.ONE_HARD);
    }
```

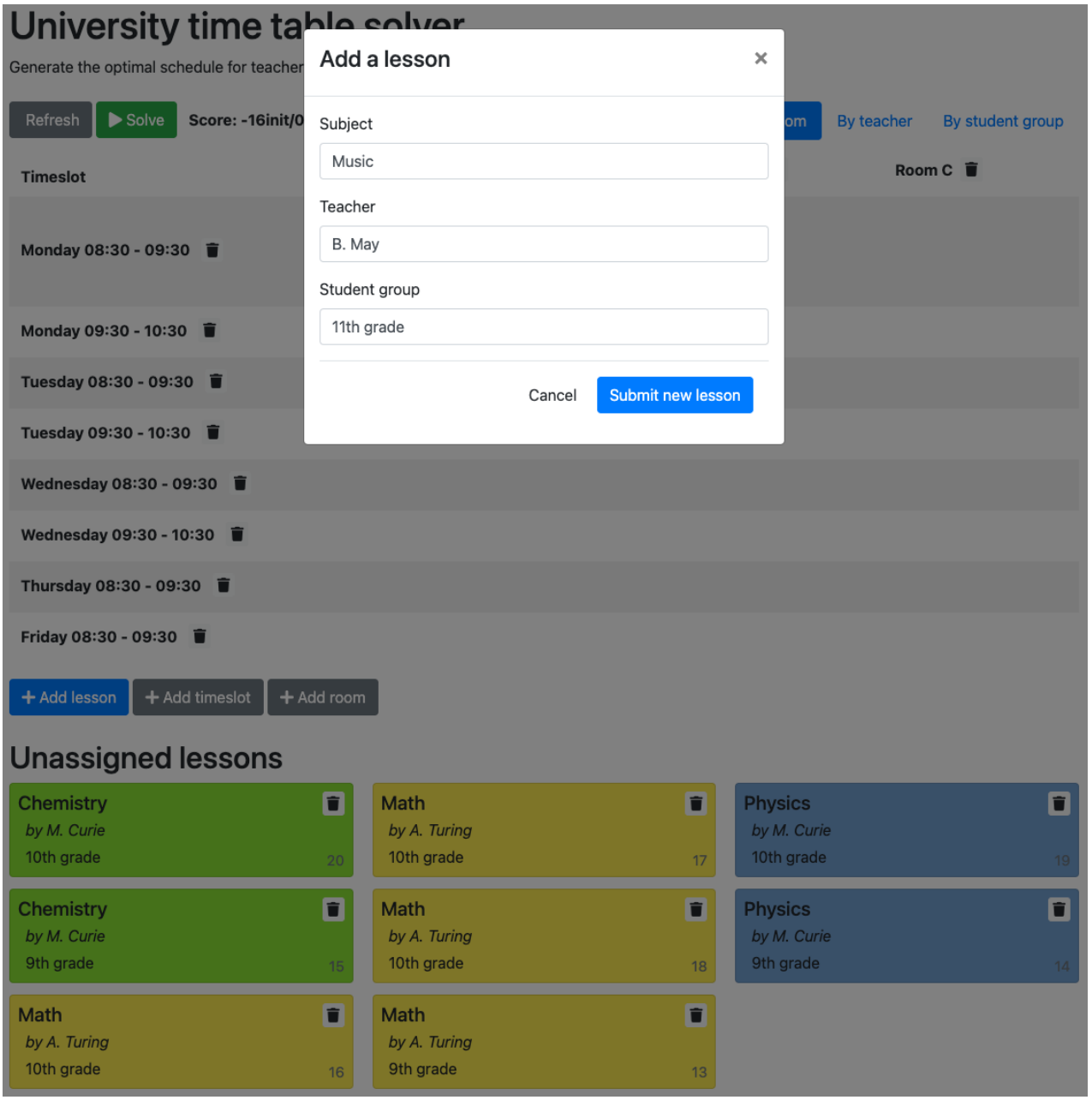
Додаток Є

Початковий інтерфейс застосунку автогенерації розкладу



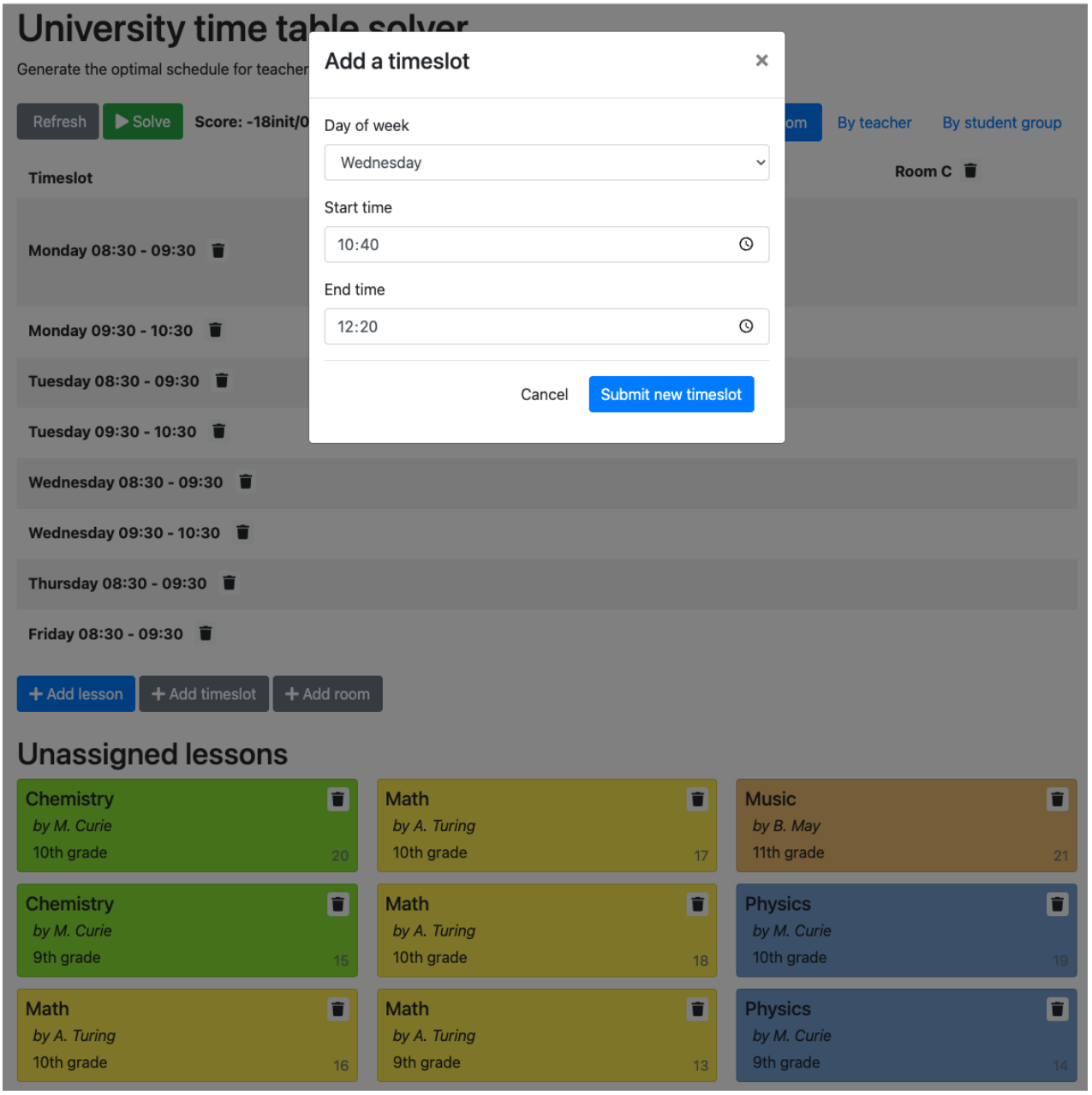
Додаток Ж

Додавання предмету до бази даних



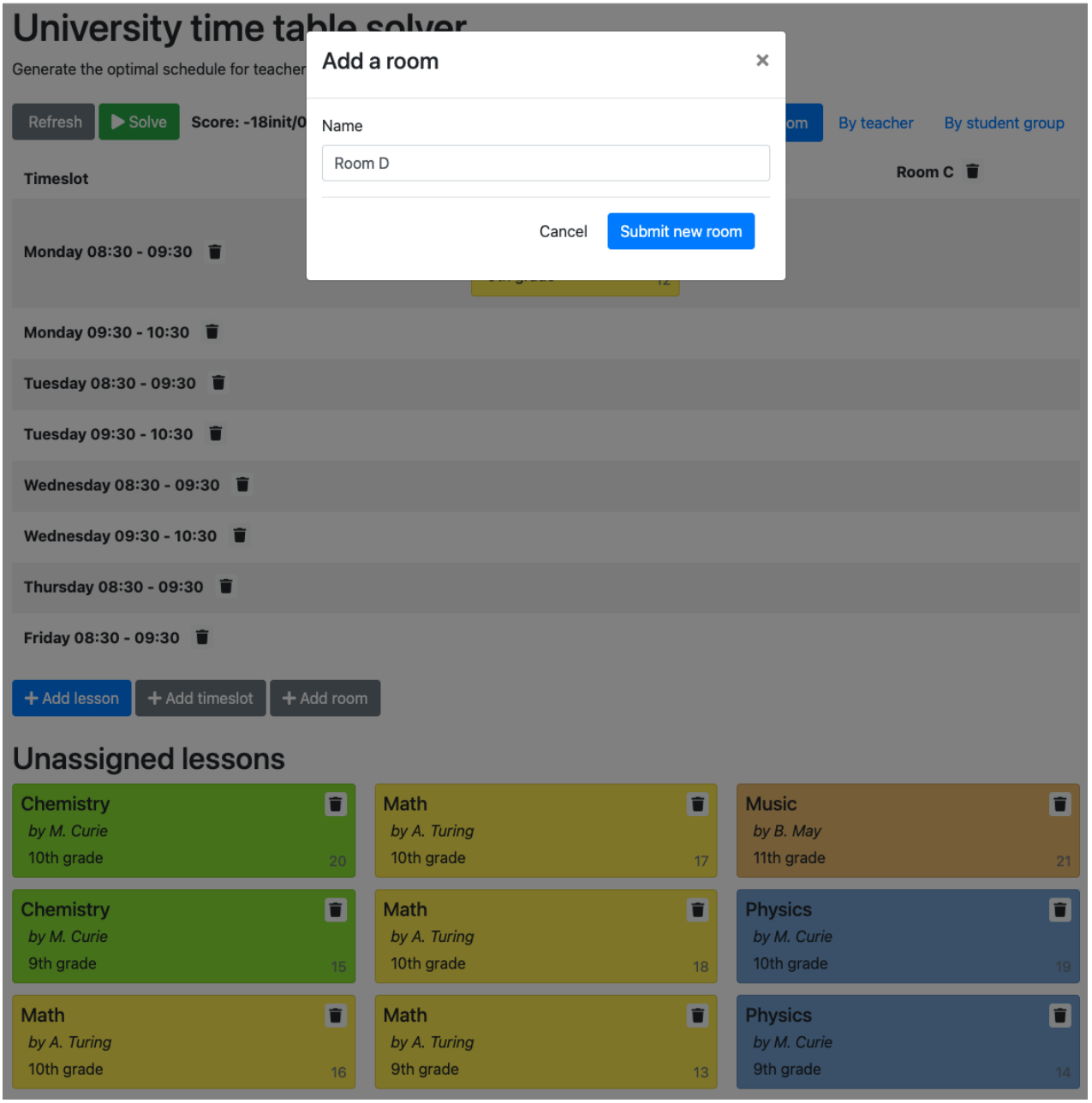
Додаток З

Додавання часового слоту, пари до бази даних



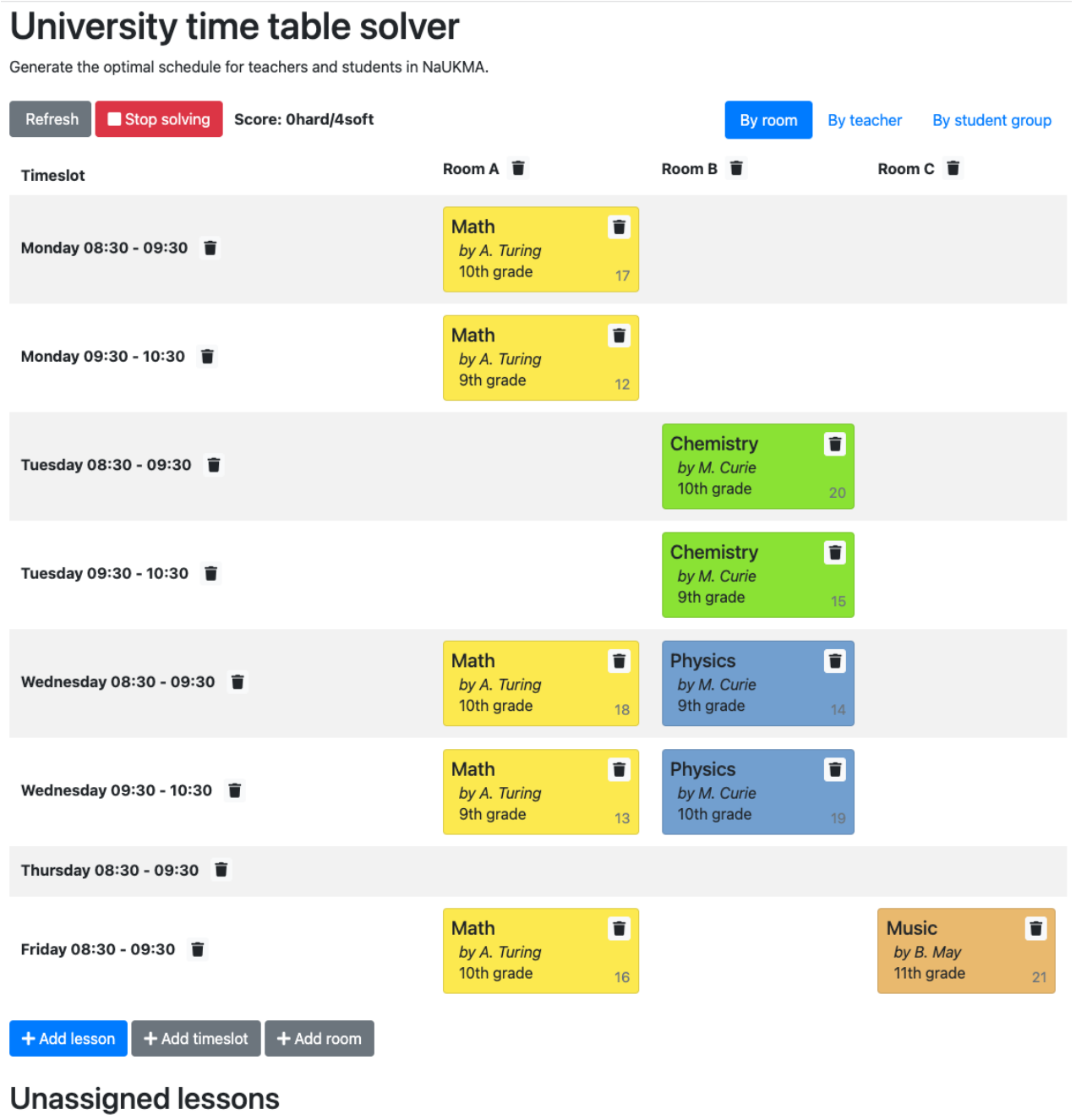
Додаток И

Додавання аудиторії до бази даних



Додаток І

Демонстрація оптимального та найкращого рішення



Додаток Ї

Демонстрація можливого, проте не достатнього рішення