КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА
Факультет комп'ютерних наук та кібернетики
Кафедра теоретичної кібернетики

Кваліфікаційна робота
на здобуття ступеня бакалавра

за спеціальністю 122 Комп'ютерні науки

на тему:

АВТОМАТИЗОВАНА ПОБУДОВА НАВЧАЛЬНИХ РОЗКЛАДІВ

Виконав студент 4-го курсу
Антон ПЕРЕХРЕСТ _____
(підпис)

Науковий керівник:
доцент кафедри теоретичної кібернетики
доцент, кандидат фіз.-мат. наук
Тетяна КАРНАУХ _____
(підпис)

Засвідчую, що в цій роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Роботу розглянуто й допущено до захисту на засіданні кафедри теоретичної кібернетики

«____»_____2022 р., протокол № ____

Завідувач кафедри
доктор фіз.-мат. наук, професор

Юрій КРАК _____

(підпис)

Київ - 2022

РЕФЕРАТ

Обсяг роботи 25 сторінок, 1 ілюстрація, 8 джерел посилань, 1 додаток.

РОЗКЛАД, ГЕНЕТИЧНИЙ АЛГОРИТМ, МЕТОД ІМІТАЦІЇ ВІДПАЛУ

Об’єктом роботи є навчальний розклад. Предметом роботи є процес побудови навчального розкладу.

Метою роботи є розроблення та програмна реалізація базового функціоналу програмного забезпечення для побудови навчального розкладу згідно заданої специфікації.

Методи розроблення: розробка програмного продукту на основі еволюційної моделі.

Інструменти розроблення: середовище розробки PyCharm, мова програмування Python.

Результати роботи: створено програмний продукт для побудови розкладів.

Створений застосунок може бути використаний у різних навчальних закладах.

ЗМІСТ

ВСТУП

Оцінка сучасного стану об’єкта розробки. Задача зі створення розкладу у навчальному закладі є типовою та відноситься до теорії розкладів — розділу науки, що займається календарним плануванням.

Створення розкладу є NP-повною задачею. Для її вирішення можна зробити недетерміновану послідовність дій, яка може вирішити задачу за поліноміальний час, але ефективних детермінованих алгоритмів досі не існує. На сьогодні й досі триває активний пошук ефективних алгоритмів для розв’язання NP-повних задач. Водночас з цим, можна намагатись будувати алгоритми, точніше підбирати евристики, що здатні більш-менш ефективно розв’язувати окремі часткові випадки таких задач.

Актуальність роботи. Основою навчального процесу є календарне планування занять у навчальному закладі. Від ефективно створеного розкладу занять залежить якість наданої освіти, яка може бути отримана в навчальному закладі. При цьому мають бути найменші затрати у вигляді викладацького складу, матеріальних витрат на функціювання навчального закладу та всього навчального закладу та всього навчального процесу. Найефективнішим чином мають бути використані матеріальні цінності навчального закладу, обладнання для проведення занять, а також аудиторні класи.

На даний час було проведено велику кількість досліджень у цій області та відокремлено найбільш ефективні з огляду на різні фактори методи та алгоритми.

Зважаючи на досить широку проблему, поставлену перед науковцями у ході створення розкладів, існує велика кількість обмежень та рекомендацій щодо календарного планування. Тому зараз теж стоїть питання над більш глибоким вивченням теми.

На етапі аналізу ринку програмних засобів було виявлено, що існують аналоги цього проєкту, проте ціна купівлі ліцензії доволі висока, також існує

проблема у тому, що програмні засоби не пристосовані до нашого випадку розробки розкладу для факультету кібернетики, тому розробка програми досить актуальна.

Мета й завдання роботи. Мета кваліфікаційної роботи полягає у створенні базового функціоналу програмного забезпечення для автоматизації процесу побудови навчального розкладу згідно заданої специфікації та з дотриманням встановлених для розкладу критеріїв якості.

Для досягнення поставленої мети необхідно виконати такі завдання:

- Розглянути різні алгоритми, що використовуються для розв’язання NP-повних задач, та вибрати найбільш придатний.
- Модифікувати алгоритм, щоб він обробляв обмеження на будований розклад.
- Розробити внутрішнє зображення вимог до розкладу та його даних.
- Виконати побудову програмного засобу, що автоматизує будову навчального розкладу.

Об’єкт, методи й засоби розроблення. Об’єктом розроблення є програмне забезпечення, що здатне автоматизувати складання навчального розкладу. Розробці передував розгляд різних підходів до розв’язання поставленої задачі. Розробка програмного продукту виконувалась на основі еволюційної моделі.

Використовувались такі інструментальні засоби, бібліотеки та технології: середовище розробки PyCharm, мова програмування Python, бібліотеки PyQt5, sqlite3, numpy.

Можливі сфери застосування. Створений застосунок може бути використаний у різних навчальних закладах та в інших випадках, що потребують складання аналогічних розкладів.

1 ОСНОВНІ ПОНЯТТЯ, ВИМОГИ ТА ОБМЕЖЕННЯ ПРИ СТВОРЕННІ РОЗКЛАДІВ

1.1 Формалізована постановка задачі. Цільова функція

Розклад — впорядкована в часі множина занять. При створенні розкладів потрібно враховувати великий масив вхідних даних та різних факторів.

Будемо вважати, що у кожному навчальному закладі є скінченна множина ресурсів, які надаються для навчального процесу. Головними ресурсами можна вважати викладацький потенціал, який має бути ефективно задіяний, матеріально- технічну базу, яка буде використана під час проведення занять різного виду, а також приміщення будівлі навчального закладу, які можуть бути використані у якості навчальних аудиторій з дотриманням усіх санітарних та інших вимог.

Головною ціллю роботи є визначення найбільш відповідного для нашого випадку алгоритму для побудови розкладу та імплементації наших досліджень у готовий до використання у реальних умовах проект, який зможе допомогти підвищенню якості освіти у навчальному закладі.

1.2 Основні поняття, використані у роботі

Поняття, які будуть використані для дослідження

Предмет.

Читається певну кількість занять на період часу.

Період часу.

Робочий 5-денний або два 5-денних тижні.

Викладач.

Читає деякий предмет для певної кількості студентів (потоку, групи чи підгрупи).

Студент або слухач.

Прослуховує предмет, який читається викладачем. Об’єднані у потоки, групи чи підгрупи.

Час заняття.

Проміжок часу, під час якого згідно розкладу відбувається заняття.

Аудиторія.

Приміщення, у якому проводиться заняття.

1.3 Основні рекомендації та обмеження щодо створення розкладу

При створенні розкладу через нашу систему маємо розглянути обмеження та рекомендації щодо календарного планування, які надаються для створення ефективного розкладу, враховуючи різні фактори.

Розклад будується для певного періоду часу (тиждень або два, тобто парні та непарні тижні, коли деякі предмети чергуються щотижнево)

- Кожна пара відбувається в один й той самий день та на одній й той самій парі за розкладом.

- Для деяких предметів, як наприклад, іноземна мова є рекомендація проведення двох пар предмету одночасно.

- Види навчальних програм на різних спеціальностях є різними, тому предмети з однаковими назвами для різних спеціальностей є різними.

- Також при складанні розкладу лекційні та практичні заняття одного й того самого предмета теж є різними.

- Для кожного предмету зіставляється множина викладачів, які мають його вести, також кожний викладач веде предмет у певній кількості об’єднань. У кожному випадку об’єднаннями є потоки, групи та підгрупи.

- Лекційні заняття проводить один викладач, у той самий час, лабораторні заняття у різних групах або підгрупах можуть проводити різні викладачі. Зазвичай між групами та підгрупами викладачів ставить адміністрація навчальної одиниці (або випадковим чином)

- Кожна аудиторія у навчальному закладі може розмістити у собі певну кількість слухачів (по деяким предметам може вказуватися окрема множина аудиторій, якщо є потреба у спеціальному лабораторному обладнанні).

- Наприклад можемо розділити таким чином:

- Лекційні класи — для проведення для цілого потоку курсу лекцій.

- Звичайні класи — меншої місткості, для проведення занять по групам та підгрупам.

- Спеціалізовані класи — для проведення занять у групах або підгрупах, коли потребується лабораторне обладнання.

- Максимальну кількість допустимої кількості пар на день у межах 4 пар встановити (для викладачів та слухачів).

- У викладачів приймемо, що максимальна кількість лекцій на день буде у межах двох пар.

- Потрібно врахувати також той чинник, що деякі викладачі з інших факультетів, тому деякі предмети можуть бути проведені у особливо визначені дні, тобто жорстко вказані, інколи ще може потребуватися жорстко вказана пара проведення.

1.4 Критерії якості

- Не повинно бути днів, коли всі пари з одного предмету.

- Не повинно бути вікон (пар, під час яких немає викладання предметів) у слухачів.

- Не повинно бути вікон у викладачів.

- Кількість пар у слухачів не повинна бути меншою, ніж дві пари на день.

- Кількість пар у викладачів не повинна бути більшою, ніж три пари на день.

Враховуючи вище вказані потреби, потрібно максимально, як це можливо побудувати ефективний розклад.

2 ЕВОЛЮЦІЙНІ АЛГОРИТМИ

Задача створення розкладів не є новою. Тому існують різні алгоритми, які були застосовані у цій проблемі. Розглянемо найбільш відповідні нам, вони є алгоритмами еволюційного виду.

Еволюційний алгоритм застосовується у штучному інтелекті, використовує моделювання біологічної еволюції. [3]

У основі лежать процеси відбору, мутації та відтворення.

Популяція — множина агентів. Поведінка популяції залежить від навколишніх умов.

Цільова функція залежить та задається довкіллям, кожен агент отримує своє значення придатності в довкіллі.

Для розмноження підходять найбільш схильні до виживання агенти. За допомогою рекомбінації та мутації агенти можуть пристосуватися до навколишніх умов.

Далі розглянемо кілька алгоритмів еволюційного виду, щоб вибрати з них найбільш оптимальний для нас.

2.1 Генетичний алгоритм

Генетичний алгоритм входить до розділу еволюційних алгоритмів пошуку. Застосовується у задачах, коли потрібно оптимізувати дані та змоделювати шляхом послідовного підбору, комбінування та варіації шуканих параметрів. У цілому алгоритм базований на різних теоріях біологічної еволюції. [2]

Для алгоритму дані потрібно представляти у формі масиву, який нагадуватиме склад хромосом. Існує початкова популяція, випадково

створюються початкові елементи у певній кількості. Потім ці елементи (особи) оцінюються функцією допасованості, якою кожній особі призначається можливість виживання. Обираються особи, які можуть схрещуватися (селекція). Далі створюється наступне покоління, за допомогою “генетичних операторів” (схрещення, мутація тощо). Наступне покоління проходить те ж саме. Утворюється еволюційний процес, продовжується, поки не буде зупинено алгоритм.

Алгоритм може бути зупинено:

- Час, даний на еволюцію закінчився.
- Число поколінь, які були дані на еволюцію закінчилося.
- Глобальне (надоптимальне) рішення було віднайдено.

Схема роботи алгоритму.

1. Початкова популяція.
2. Схрещування і (або) мутація.
3. Селекція.
4. Формування нового покоління.

2.2 Бджолиний алгоритм

Алгоритм застосовується для знаходження глобальних екстремумів складних функцій. [5]

Сам алгоритм імітує поведінку бджіл, коли вони зграєю харчуються.

На відміну від генетичних алгоритмів, які беруть в основу процес природного відбору та еволюції, бджолиний використовує поведінку бджіл у соціумі. Існує можливість у об’єднанні цих методів, так як у бджолиному легше знайти точки глобального мінімуму

Розберемо роботу даного алгоритму.

Існує зграя бджіл. Для існування їм потрібно опилювати квіти. Тому

потрібно знайти галявину, де буде найбільша кількість квітів. Бджоли випадково шукають початкове місце, з випадковою швидкістю льоту. Бджоли можуть запам’ятовувати місця, де було віднайдено найбільшу кількість квітів та порівнювати з місцями, знайденими іншими бджолами. Потім у бджоли є вибір або повернутися до найбільш вигідного для неї місця, або знайти нове місце, яке було передано іншими зі зграї. Вона летить туди, куди вирішить найбільш важливим — місце, де вона була зі своєї пам’яті, або те, яке їй порадили — соціальний рефлекс. По дорозі може бути знайдене інше найбільш вигідне місце. Вони постійно перевіряють місця під час льоту з попередніми. Зупиняється у найбільш наповненому квітами місці, потім й інші бджоли тут осідають.

2.3 Мурашиний алгоритм

Мурашиний алгоритм використовується для знаходження наближених розв’язків задачі комівояжера та задачі пошуку маршрутів на графах. [4]

Алгоритм побудований на поведінці мурашиної колонії. Мурахи під час походу маркують найбільш вигідні шляхи феромоном.

Починається з розміщення мурах у певних містах (вершинах графу), далі ймовірнісним методом рухаються далі. Мурахи фіксують свої позиції та прийняті рішення, згодом наступний похід дає кращі результати. Бджолиний алгоритм є одним з варіантів цього підходу.

Результат неточний, проте при багаторазовому повторенні алгоритму може бути отримане досить точне наближення.

2.4 Алгоритм імітації відпалу та еволюційні алгоритми

Алгоритм імітації відпалу відноситься до низки методів Монте-Карло та використовується для розв'язання глобальної задачі оптимізації. Використовується зазвичай, коли знаходження глобального оптимуму є важливішим, ніж знаходження локального оптимуму. Сама суть алгоритму йде від фізичних процесів. [1]

Алгоритм використовує поняття температури: за високої температури атоми металу більш рухливі та порушують просторову структуру, але за зменшення температури їх геометричне розташування отримує все більш та більш однорідну структуру.

Основна ідея алгоритму полягає в тім, що, маючи ітераційний процес, у випадку, коли поточна ітерація дає гірше значення ніж було на попередній, з певною ймовірністю ми приймаємо рішення перейти до отриманого гіршого значення. І чим вищою буде температура, тим вищою буде ймовірність такого переходу.

Зрозуміло, що така схема передбачає елемент випадковості в отриманні нового значення. Серед розглянутих еволюційних алгоритмів для отримання нового значення доволі непогано підійде генетичний алгоритм.

3 РЕАЛІЗАЦІЯ ПРОЄКТУ

3.1 Огляд використаних інструментів

Для розробки програмного комплексу основною мовою програмування було обрано Python. [8]

Python – інтерпретована об’єктно-орієнтована мова програмування високого рівня.

За допомогою даної мови програмування можна легко підключатися та працювати з базами даних та SQL у цілому. Основною IDE для розробки було обрано PyCharm через простоту та ефективність використання.

3.2 Вхідні дані програми

Вхідні дані програми, за якими необхідно побудувати розклад, містяться в реляційній базі даних. У проєкті використано sqlite3, як кросплатформену вбудовану вільно доступну базу даних.

Зокрема, у використовуваній базі даних містяться такі дані

- ID групи
- Кількість студентів
- Назва групи
- Спеціальність
- ID дисципліни
- Назва дисципліни
- Тип дисципліни
- Курс
- Викладач

- Ступінь викладача

- Аудиторія

- Тип аудиторії

- Місткість аудиторії

Ці дані зберігаються у вигляді таблиць. Нижче наведено схему використовуваної бази даних.

```
CREATE TABLE IF NOT EXISTS "Specialties" (
     "Specialty_ID"     INTEGER NOT NULL UNIQUE,
     "Specialty_Name"   TEXT NOT NULL UNIQUE,
     PRIMARY KEY("Specialty_ID" AUTOINCREMENT)
);
CREATE TABLE IF NOT EXISTS "Teacher_Position" (
     "Position_ID" INTEGER NOT NULL UNIQUE,
     "Position"    TEXT NOT NULL UNIQUE,
     PRIMARY KEY("Position_ID" AUTOINCREMENT)
);
CREATE TABLE IF NOT EXISTS "Teachers" (
     "Teacher_ID"  INTEGER NOT NULL UNIQUE,
     "Teacher_Name"     TEXT NOT NULL,
     "Position_ID" INTEGER NOT NULL,
     PRIMARY KEY("Teacher_ID" AUTOINCREMENT)
);
CREATE TABLE IF NOT EXISTS "Classroom_Type" (
     "Type_ID"INTEGER NOT NULL UNIQUE,
     "Type"   TEXT NOT NULL UNIQUE,
     PRIMARY KEY("Type_ID" AUTOINCREMENT)
);
CREATE TABLE IF NOT EXISTS "Classrooms" (
     "Classroom_ID"     INTEGER NOT NULL UNIQUE,
     "Class_Number"     INTEGER NOT NULL UNIQUE,
     "Number_Of_Seats"  INTEGER NOT NULL,
     "Type_ID"INTEGER NOT NULL,
     PRIMARY KEY("Classroom_ID" AUTOINCREMENT),
     FOREIGN KEY("Type_ID") REFERENCES
"Classroom_Type"("Type_ID")
);
CREATE TABLE IF NOT EXISTS "Discipline_Type" (
     "Disc_Type_Id"     INTEGER NOT NULL UNIQUE,
     "Disc_Type_Name"   TEXT NOT NULL UNIQUE,
     PRIMARY KEY("Disc_Type_Id" AUTOINCREMENT)
);
CREATE TABLE IF NOT EXISTS "Discipline_List" (
     "ID" INTEGER NOT NULL UNIQUE,
```

```
    "Discipline_ID"    INTEGER NOT NULL,
    "Disc_Type_ID"     INTEGER NOT NULL,
    "Group_ID"    INTEGER NOT NULL,
    "Specialty_ID"     INTEGER NOT NULL,
    "Year"    INTEGER NOT NULL,
    "Teacher_ID"  INTEGER NOT NULL,
    PRIMARY KEY("ID" AUTOINCREMENT),
    FOREIGN KEY("Disc_Type_ID") REFERENCES
"Discipline_Type"("Disc_Type_Id"),
    FOREIGN KEY("Teacher_ID") REFERENCES
"Teachers"("Teacher_ID"),
    FOREIGN KEY("Discipline_ID") REFERENCES
"Disciplines"("Discipline_ID"),
    FOREIGN KEY("Group_ID") REFERENCES
"Student_Groups"("Group_ID"),
    FOREIGN KEY("Specialty_ID") REFERENCES
"Specialties"("Specialty_ID")
);
CREATE TABLE IF NOT EXISTS "Disciplines" (
    "Discipline_ID"    INTEGER NOT NULL UNIQUE,
    "Discipline_Name"  TEXT NOT NULL UNIQUE,
    PRIMARY KEY("Discipline_ID" AUTOINCREMENT)
);
CREATE TABLE IF NOT EXISTS "Student_Groups" (
    "Group_ID"    INTEGER NOT NULL UNIQUE,
    "Amount_Of_Students"    INTEGER NOT NULL,
    "Group_Name"  TEXT NOT NULL,
    "Specialty"   INTEGER,
    PRIMARY KEY("Group_ID" AUTOINCREMENT),
    FOREIGN KEY("Specialty") REFERENCES
"Specialties"("Specialty_ID")
);
```

3.3 Завантаження вхідних даних

Всі дані, які ми початково беремо для дослідження знаходяться у базі даних.

Дану базу даних підключаємо до нашого проекту. Це найбільш оптимальний варіант. Проте, для того, щоб оптимізувати введення у програму даних базу даних тому модифікуємо під потреби нашої програми у проекті. У базі даних повністю тримається у собі всі дані, про студентів та викладачів, предмети, та в цілому про факультет, спеціальності, курси та групи. [7]

Поступово з бази даних вся інформація починає використовуватися у нашому алгоритмі.

Для підключення до бази даних і отримання з неї даних використовується такий програмний код.

```
with connection:
    cursor = connection.cursor()
    cursor.execute("SELECT * FROM Discipline_List")
    disciplines = cursor.fetchall()
    cursor.execute("SELECT * FROM Disciplines")
    discipline_names = cursor.fetchall()
    cursor.execute("SELECT * FROM Discipline_Type")
    discipline_types = cursor.fetchall()
    cursor.execute("SELECT * FROM Teachers")
    teachers = cursor.fetchall()
    cursor.execute("SELECT * FROM Teacher_Position")
    teacher_positions = cursor.fetchall()
    cursor.execute("SELECT * FROM Specialties")
    specialties = cursor.fetchall()
    cursor.execute("SELECT * FROM Student_Groups")
    groupsData = cursor.fetchall()
    group_amount = len(groupsData)
    groups = []
    for gr in groupsData:
        groups.append(gr[0])
```

3.4 Реалізація алгоритму імітації відпалу

Перш за все, наявна цільова функція `aim_limitation`, що оцінює якість розкладу: чим більше вимог порушено, тим більше значення функції.

- Задаємо початкову температуру та ітерації.
- Створюємо початковий набір розкладів випадковим чином.
- Обчислюємо значення цільової функції для отриманих розкладів
- Випадковий розклад змінюємо:
    - Випадково виділяємо частину фрагментів, яка буде змінена
    - Перевірка на обмеження

- Рахуємо функції для нового розкладу та порівнюємо з початковим, обираємо кращий

- Потрібно визначити ймовірність для застосування розподілу Гіббса за результатом обираємо знову кращий розклад

- Поступово змінюємо температуру у меншу сторону до досягнення найкращого результату.

Текст файлу з кодом реалізації наведено в Додатку А. Основною функцією «відпалу» є функція annealing.

У програмі зроблено виконання обмежень, а саме:

- Немає днів, коли всі пар з одного предмету

- При кількості груп більше двох немає вікон у слухачів

- При кількості груп більше двох немає вікон у слухачів

- При кількості груп більше двох кількість пар на день у слухачів не менше двох пар на день.

- Кількість пар у викладачів не більше трьох на день.

Результат роботи програми наведений на рис. 1.

Розклад за імітацією відпалу

	1	2	3
1	Група	ТК-41	ТК-42
2		Понеділок	
3	Пара 1	Коректність програм та логіки програмування Лекція Федорус О. М. Асистент	Коректність програм та логіки програмування Лекція Федорус О. М. Асистент
4	Пара 2	Розробка бізнес аналітичних систем Лекція Вергунова І. М. Професор	Розробка бізнес аналітичних систем Лекція Вергунова І. М. Професор
5	Пара 3	Нейронні мережі та нейрообчислення Лекція Пашко А. О. Професор	Нейронні мережі та нейрообчислення Лекція Пашко А. О. Професор

Рисунок 1 — Результат роботи програми

3.5 Розгортання застосунку

Програмно-апаратні вимоги

Для використання даної програми має бути встановлений інтерпретатор Python 3, а також бібліотеки numpy, sqlite3, PyQt5.

Інструкція для користувача

Для того щоб почати користуватися програмою, треба внести дані про необхідний розклад у базу даних (файл ScheduleUniv1.db). Точкою входу в програму є файл startSchedule.py.

Кожний раз після запуску формується новий розклад з дотриманням поставлених нами обмежень. На даному етапі розробки проєкту немає можливості зберегти розклад у певний файл, для подальшого редагування. Але у майбутніх реалізаціях проєкту дана можливість буде додана.

ВИСНОВКИ

У роботі було:

— проаналізовано актуальність створення розкладів у вищих навчальних закладах;

— виконано аналіз можливих методів та алгоритмів для створення розкладів;

— описано початкові дані для дослідження теми;

— створено працюючу модель застосунку системи для автоматизації побудови навчальних розкладів у навчальних закладах на основі реалізації алгоритму імітації відпалу;

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Алгоритм імітації відпалу [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC_%D1%96%D0%BC%D1%96%D1%82%D0%B0%D1%86%D1%96%D1%97_%D0%B2%D1%96%D0%B4%D0%BF%D0%B0%D0%BB%D1%83.

2. Генетичні алгоритми: суть, опис, приклади, застосування [Електронний ресурс] – Режим доступу до ресурсу: https://presa.com.ua/aktualne/genetichni-algoritmi-sut-opis-prikladi-zastosuvannya.html

3. Еволюційний алгоритм [Електронний ресурс] – Режим доступу до ресурсу: https://www.wikiwand.com/uk/%D0%95%D0%B2%D0%BE%D0%BB%D1%8E%D1%86%D1%96%D0%B9%D0%BD%D0%B8%D0%B9_%D0%B0%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC

4. Мурашиний алгоритм [Електронний ресурс] – Режим доступу до ресурсу: https://www.wikizero.com/uk/%D0%9C%D1%83%D1%80%D0%B0%D1%88%D0%B8%D0%BD%D0%B8%D0%B9_%D0%B0%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC

5. Бджолиний алгоритм [Електронний ресурс] – Режим доступу до ресурсу: https://www.wikiwand.com/uk/%D0%91%D0%B4%D0%B6%D0%BE%D0%BB%D0%B8%D0%BD%D0%B8%D0%B9_%D0%B0%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC

6. Теорія розкладів [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/%D0%A2%D0%B5%D0%BE%D1%80%D1%96%D1%8F_%D1%80%D0%BE%D0%B7%D0%BA%D0%BB%D0%B0%D0%B4%D1%96%D0%B2

7. SQL using Python [Електронний ресурс] – Режим доступу до ресурсу:

https://www.geeksforgeeks.org/sql-using-python/

8. Python Tutorial

[Електронний ресурс] – Режим доступу до ресурсу:

https://www.w3schools.com/python/

ДОДАТОК А

```
import copy
import numpy as np
from math import *
from random import *
import scheduleData as data


def nel(roz, change_percent):
    schedule = copy.copy(roz)
    nm = int(round(change_percent * len(schedule) / 100))
    nuse = list(range(0, len(schedule)))
    while nm > 0:
        ind = choice(nuse)
        nuse.remove(ind)
        group = []
        for d in schedule[ind]['discipline']:
            group.append(data.disciplines[d - 1][3])
        all_positions_notEmpty,
all_positions_notEmptyWithGroups,
all_positions_emptyWithGroups\
        = data.all_positions(schedule, group, 0, 0, 1, 1, 0,
1)
        if len(all_positions_emptyWithGroups) > 0:
            empty = choice(all_positions_emptyWithGroups)
            all_positions_emptyWithGroups.remove(empty)

all_positions_emptyWithGroups.append([schedule[ind]['day'],
schedule[ind]['lesson'], group])
            schedule[ind]['day'] = empty[0]
            schedule[ind]['lesson'] = empty[1]
        elif len(all_positions_notEmptyWithGroups) > 0:
            notEmpty =
choice(all_positions_notEmptyWithGroups)
            all_positions_notEmptyWithGroups.remove(notEmpty)
            temp = schedule[ind]
            indNotEmpty =
all_positions_notEmpty.index(notEmpty)
            schedule[ind]['day'] = notEmpty[0]
            schedule[ind]['lesson'] = notEmpty[1]
            schedule[indNotEmpty]['day'] = temp['day']
            schedule[indNotEmpty]['lesson'] = temp['lesson']
            nm -= 1
def annealing(iter_num, begin_num):
    T = 1000
    population = data.create_population(begin_num)
    mpop = []
```

```
for element in population:
    mpop.append(data.aim_limitation(element))
for i in range(iter_num):
    if T == 0:
        break
    index = randint(0, begin_num - 1)
    sch = nel(population[index], randint(0, 101))
    ll = data.aim_limitation(sch)
    if mpop[index] <= ll:
        population[index] = sch
        mpop[index] = ll
    else:
        delta = mpop[index] - ll
        p = exp((-1) * delta / T)
        q = np.random.uniform(0, 1, 1)[0]
        if p > q:
            population[index] = sch
            mpop[index] = ll
        if i != 0:
            T = T / log(i + 1)
print("max=", max(mpop))
return population, population[mpop.index(max(mpop))]
    else:
        nm += 1
    nm -= 1
return schedule
```