Проект по курсу «Компьютерные технологии»

.

Составление и оптимизация расписания

Преподаватель:
Бабичев Сергей Леонидович

Выполнили:
Гавриленко Арсений, 777
Киселев Кирилл, 778
Постников Григорий, 777
Самойленко Александр, 778
Усеинов Булат, 778

МФТИ
Долгопрудный
23 мая, 2019

A. Составление расписания

Цель: Генерация оптимального, в смысле некоторого критерия, расписания.

На вход:

- Список групп: < `Имя группы, количество учащихся, доступное время` >
- Список преподавателей: < `Преподаватель, доступное время` >
- Список кабинетов: < `Кабинет, вместимость, допустимое время` >
- Список предметов: < `Предмет, тип, преподаватель, длительность, группы`>

На выход:

- Распределенные события:

  < `День недели, номер пары, предмет, преподаватель, кабинет` >

Генерация:

Задача составления расписания является NP-полной. Решать ее предлагается следующим образом: рассматривается предмет, который ведет преподаватель с наименьшим допустимым временем и ставится в расписание (в этом процессе используется поиск с возвратом - backtracking). По итогу генерируется произвольное корректное расписание. Корректность подразумевает за собой отстутствие наложений пар, наличие возможности у соответствующего преподавателя провести каждую сгенерированную пару в свободном в это время кабинете, а также, что, безусловно, важно, соответствие учебной программе курса.

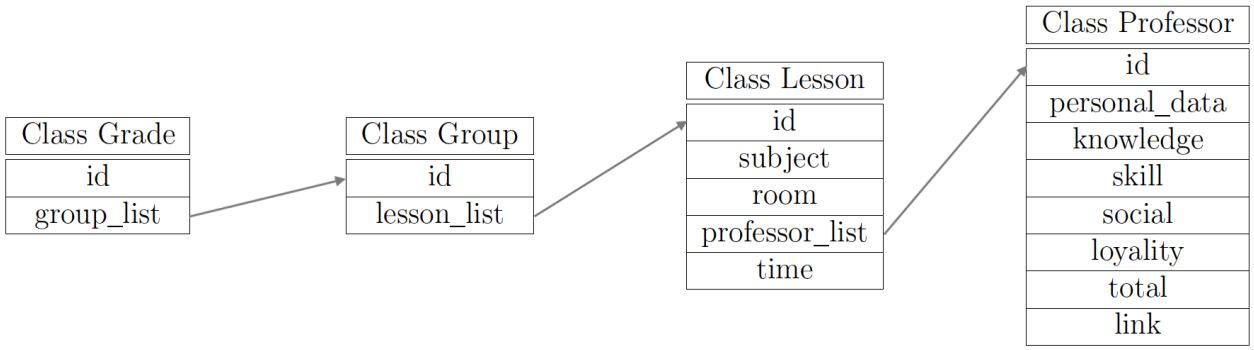
Изменение:

Сгенерированное расписание удовлетворяет только самому главному из требований: возможности провести каждую сгенерированную пару. Очевидно, что данное расписание не всегда является наиболее оптимальным для среднего студента и преподавателя. Для примера можно выделить случай, когда несколько дней подряд полностью загруженны парами, а оставшиеся на неделе пары - свободны, или когда во все дни пары начинаются не с первой, а со второй или третьей. Для того, чтобы этого избежать предлагается оптимизировать некоторый функционал. В нашем решении этой задачи использовался следующий: берется *сумма порядковых номеров семинаров + удвоенная сумма номеров лекций*, что, во-первых, позволяет получить более раннее расположение семинаров и лекций, а во-вторых, влияет на более равномерное распределение пар по дням. В качестве метода оптимизации использовался алгоритм искусственной колонии пчел.

Б. Оптимизация существующего расписания

Цель: Имея некоторое начальное расписание, модифицировать его таким образом, чтобы оно наилучшим образом удовлетворяло критериям пользователя.

Структура данных:



- Class Grade содержит список групп *group_list*, которые существуют на курсе, соответствующем номеру *id*.
- Class Group содержит список пар *lesson_list*, которые числятся у группы с номером *id*.
- Class Lesson содержит название пары - *subject*, кабинет *room*, список преподавателей *professor_list* (может быть несколько преподавателей, например, в случае пары по физическому практикуму, поэтому храним их в списке), время проведения пары с порядковым номером *id*, которое меняется от 0 до 41 (порядковый номер пары в неделе).
- Class Professor содержит личные данные преподавателя - *personal_data*: имя, фамилия и отчество; рейтинги, взятые с ресурса wikimipt.org, а также уникальный номер, присвоенный каждому преподавателю.
- Также классы содержат некоторые функции, упрощающие работу с ними, например для наглядной визуализации информации, функций, проводящие модификации объектов класса, и т.д.

Что оптимизировать?:

Пользователю предлагаются на выбор следующие параметры, которые могут играть роль при составлении нового расписания:

- Рейтинги преподавателей.
- Количество окон.
- Освобождение дней или конкретных пар.
- Время обучения в конкретные дни: раньше начинать, позже заканчивать и т.д.

Для каждого из выбранных параметров пользователь с помощью ползунка в соответствующем окне выбирает вес того, насколько важен ему этот параметр. Эти данные передаются в виде числа от 0 до 1 и передаются на сервер, где происходят расчеты.

Функция потерь:

$$L = w_1 \cdot (1 - \frac{\text{рейтинг}}{\text{max}}) + w_2 \cdot \frac{\text{кол-во окон}}{\text{кол-во пар}} +$$

$+w_3 \cdot (\text{штраф за позднее начало}) + (\text{штрафы за наличие пар в запрещенное время})$

Оптимизация:

В качестве метода оптимизации используется имитация отжига. На каждой новой итерации рассматривается новое расписание, отличающееся от предыдущего заменой двух пар. Если значение функции потерь на новом расписании ниже, чем на старом, то оно берется в качестве основного, если же выше, то оно берется в качестве основного с некоторой вероятностью, которая экспоненциально убывает с номером итерации и разницей между значениями функций потерь. Таким образом, по завершении работы алгоритма будет выдано расписание с низким (относительно других расписаний) показателем функции потерь.

Вывод:

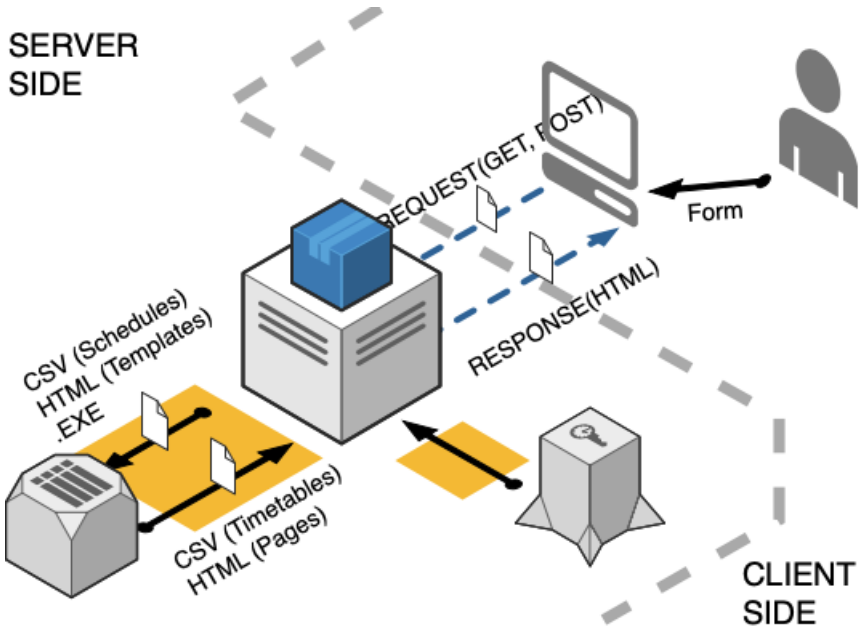
По окончании работы алгоритма выведется окно, в котором с одной стороны находится расписание до оптимизации, а с другой стороны - предложенное новое, которое лучше подходит под заданные критерии. Таким образом, пользователь наглядно увидит все изменения, которые ему следует провести со своим расписанием, чтобы оно его больше удовлетворяло.

B. Сервер

Взаимодействие клиента с сервером происходит в несколько этапов:

1. Клиент подключается к серверу и ждет ответ в виде html-страницы.
2. После получения этих данных браузер пользователя показывает ему главную страницу проекта Optimize.
3. Далее пользователь может нажать кнопку "Optimize"и попасть на страницу оптимизации.
4. В таком случае сервер посылает пользователю форму, в которой ему необходимо указать свои требования к новому расписанию. После заполнения нажимается submit-клавиша (также имеющая название "Optimize"), и данные отправляются на сервер.
5. Выполняется обработка полученных данных и запускается алгоритм, написанный на C++.
6. Далее на сервере происходит обработка выходных данных алгоритма, собирается html-страница ответа и отправляется пользователю.
7. На экране пользователя появляется новое расписание.

Работа сервера схематично показана на диаграмме:



Г. О проекте

Технологии и библиотеки:

Для обеспечения работоспособности проекта использовалось следующее:
C++, Python3, Django, Subprocess, Pandas, Numpy, Csv, Bootstrap4, JavaScript.

Распределение обязанностей:

- Гавриленко Арсений - системный дизайн, все структуры данных и обработка входных и выходных данных. Расширение по типу Pandas на C++.

- Киселев Кирилл - реализация имитации отжига для решения данной задачи, разработка, написание алгоритма решения задачи и метрики, позволяющей оценивать качество расписания, приведение необработанных данных к необходимому виду.

- Самойленко Александр - реализация имитации отжига для решения данной задачи, разработка и написание алгоритма решения задачи, парсинг данных с сайта wikimipt.org, приведение необработанных данных к необходимому виду.

- Постников Григорий - реализация алгоритма искусственной колонии пчел для реализации задачи составления расписания с чистого листа, приведение необработанных данных к необходимому формату, а также проектирование архитектуры хранилища данных.

- Усеинов Булат - Web-разработка, установка клиент-серверного взаимодействия, UX/UI, пред и постобработка данных клиента.