|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

**РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА ПО ПРОИЗВОДСТВЕННОЙ ПРАКТИКЕ**

Студент\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Нгуен Санг Фыок\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*фамилия, имя, отчество*

Группа \_\_\_\_ИУ7И-66Б\_\_\_\_\_\_

Название предприятия \_\_\_\_\_\_МГТУ им. Н. Э. Баумана, каф. ИУ7\_\_\_\_\_\_\_\_\_\_\_\_

Студент **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_** **Нгуен С. Ф**.\_**\_\_\_\_**

*подпись, дата фамилия, и.о.*

Руководитель курсового проекта **\_\_\_\_\_\_\_\_\_\_\_\_Толпинская Н. Б.\_\_\_\_\_**

*подпись, дата фамилия, и.о.*

Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*Москва, 2021 г.*

|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

УТВЕРЖДАЮ

Заведующий кафедрой ИУ7

(Индекс)

\_\_\_\_\_\_\_\_\_\_\_\_\_ И.В.Рудаков

(И.О.Фамилия)

« \_\_\_ » \_\_\_\_\_\_\_\_\_\_\_\_ 2021 г.

**ЗАДАНИЕ**

**на выполнение производственной практики**

Разработка программы распределения нагрузки и учебных групп по преподавателям

(Тема производственной практики)

Студент 1. Нгуен Ф. С. гр. ИУ7-66Б

2. Нгуен Т. Т. гр. ИУ7-66Б

3. Нгуен А. Т. гр. ИУ7-66Б

(Фамилия, инициалы, индекс группы)

1. Техническое задание

Разработка программы распределения нагрузки и учебных групп по преподавателям, с учетом ограничения на объем часов - нагрузку преподавателя

***2. Оформление производственной практики***

Расчетно-пояснительная записка на 25-30 листах формата А4.

Расчетно-пояснительная записка должна содержать постановку введение, аналитическую часть, конструкторскую часть, технологическую часть, экспериментально-исследовательский раздел, заключение, список литературы, приложения.

Дата выдачи задания « \_\_\_\_ » \_\_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_ г.

**Руководитель производственной практики** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Толпинская Н. Б.

(Подпись, дата) (И.О.Фамилия)

**Студент** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**\_\_ \_\_\_** **\_** Нгуен Ф. С.

(Подпись,дата) (И.О.Фамилия)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**\_\_ \_\_\_** Нгуен Т. Т.

(Подпись,дата) (И.О.Фамилия)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**\_\_ \_\_\_** Нгуен А. Т.

(Подпись,дата) (И.О.Фамилия)

**Оглавление**

[Введение 4](#_Toc83640507)

[**1.** **Аналитическая часть** 5](#_Toc83640508)

[**1.1.** **Генетический алгоритм** 5](#_Toc83640509)

[**1.2.** **Задача распределения нагрузки и учебных групп (задача составления расписания занятий)** 8](#_Toc83640510)

[**1.3.** **Решение задачи на основе генетического алгоритма** 9](#_Toc83640511)

[***1.3.1.*** ***Хромосома*** 9](#_Toc83640512)

[***1.3.2.*** ***Функция фитнеса*** 12](#_Toc83640513)

[**1.3.3.** **Кроссовер** 12](#_Toc83640514)

[**1.3.4.** **Мутация** 13](#_Toc83640515)

[**1.3.5.** **Алгоритм** 13](#_Toc83640516)

[**2.** **Конструкторская часть** 14](#_Toc83640517)

**[2.1.](#_Toc83640518)****[Выбор используемых типов и структур данных](#_Toc83640518)** [14](#_Toc83640518)

**[2.2.](#_Toc83640519)****[Структура и состав классов](#_Toc83640519)** [14](#_Toc83640519)

[**3.** **Технологическая часть** 15](#_Toc83640520)

[3.1. **Выбор и обоснование языка программирования и среды разработки** 15](#_Toc83640521)

[**3.2.** **Сведения о модулях программы** 15](#_Toc83640522)

[**3.3.** **Интерфейс программы** 15](#_Toc83640523)

[**4.** **Экспериментальная часть** 17](#_Toc83640524)

[4.1. **Цель эксперимента:** 17](#_Toc83640525)

[**5.** **Заключение** 18](#_Toc83640526)

[**6.** **Список использованной литературы** 19](#_Toc83640527)

# Введение

Составление расписания занятий - одна из тех сложных задач NP. Проблема может быть решена с помощью алгоритма эвристического поиска для поиска оптимального решения, но он работает только для простых случаев. Для более сложных входных данных и требований поиск достаточно хорошего решения может занять некоторое время или может оказаться невозможным. Здесь в игру вступают генетические алгоритмы.

Целью проекта является: реализовать построение трехмерной городской сцены.

Чтобы достигнуть поставленную цель, требуется решить следующие задачи:

1. изучение алгоритма Генети́ческого;
2. разработка базы данных;
3. реализация и применение алгоритма Генети́ческого для задачи распределения нагрузки и учебных групп;
4. разработка пользовательского интерфейса
5. **Аналитическая часть**

В прошлом для решения всех оптимальных задач люди часто использовали классические методы, такие как восхождение на холмы, моделирование сталеплавильного производства... С проблемами с небольшими пространствами поиска. Тогда вышеуказанные методы могут хорошо решить эту проблему. Но с большим пространством поиска вышеперечисленные методы неэффективны. Следовательно, это условие требует от нас новых методов, позволяющих хорошо решать эти проблемы. В настоящее время для решения задачи оптимизации мы можем использовать «генетический алгоритм».

* 1. **Генетический алгоритм**

Генети́ческий алгори́тм (genetic algorithm) — это эвристический алгоритм поиска, используемый для решения задач оптимизации и моделирования путём случайного подбора, комбинирования и вариации искомых параметров с использованием механизмов, аналогичных естественному отбору в природе. Является разновидностью эволюционных вычислений, с помощью которых решаются оптимизационные задачи с использованием методов естественной эволюции, таких как наследование, мутации, отбор и кроссинговер. Отличительной особенностью генетического алгоритма является акцент на использование оператора «скрещивания», который производит операцию рекомбинации решений-кандидатов, роль которой аналогична роли скрещивания в живой природе.

Естественный отбор: Процесс естественного отбора начинается с отбора наиболее приспособленных особей из популяции. Они производят потомство, которое наследует характеристики родителей и будет добавлено к следующему поколению. Если родители имеют лучшую физическую форму, их потомство будет лучше родителей и будет иметь больше шансов на выживание. Этот процесс продолжается итерация, и в конце будет найдено поколение с наиболее приспособленными людьми.

В генетическом алгоритме рассматриваются пять этапов : Начальная популяция. Функция фитнеса, Выбор, Кроссовер, Мутация

1. **Начальная популяция**

Процесс начинается с набора индивидов, который называется Популяцией. Каждый человек - это решение проблемы, которую вы хотите решить.

Индивидуум характеризуется набором параметров (переменных), известных как гены. Гены объединяются в цепочку, образуя хромосому (раствор).

В генетическом алгоритме набор генов человека представлен в виде строки в виде алфавита. Обычно используются двоичные значения (строка из единиц и нулей).

1. **Функция фитнеса**

Функция фитнеса определяет, насколько человек физически подготовлен (способность человека соревноваться с другими людьми). Он дает оценку пригодности для каждого человека. Вероятность того, что особь будет выбрана для воспроизводства, основана на ее оценке пригодности.

1. **Выбор**

На этом этапе отбираются особи для воспроизводства потомства. Затем отобранные особи распределяются попарно по двое для улучшения воспроизводства. Эти люди передают свои гены следующему поколению.

Основная цель этого этапа - создать регион с высокими шансами на создание лучшего решения проблемы (лучше, чем в предыдущем поколении). Генетический алгоритм использует метод пропорционального отбора по пригодности, чтобы гарантировать, что полезные решения используются для рекомбинации.

1. **Кроссовер**

Кроссовер - самый важный этап в генетическом алгоритме. Для каждой пары родителей, подлежащих спариванию, точка кроссовера выбирается случайным образом изнутри генов.

На этом этапе создается детское население. В алгоритме используются операторы вариации, которые применяются к родительской популяции.

Потомство создается путем обмена генами родителей между собой, пока не будет достигнута точка кроссовера. Новое потомство добавляется к популяции.

1. **Мутация**

В некоторых новых потомках некоторые из их генов могут подвергаться мутации с низкой случайной вероятностью. Это означает, что некоторые биты в битовой строке могут быть перевернуты. Мутация происходит для поддержания разнообразия в популяции и предотвращения преждевременной конвергенции.

Население имеет фиксированный размер. По мере формирования новых поколений люди с наименьшей физической подготовкой умирают, предоставляя место для нового потомства.

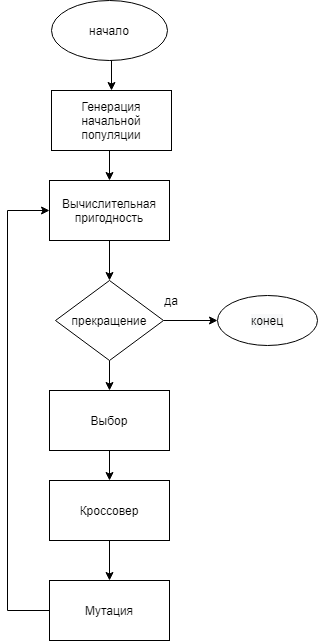
Последовательность фаз повторяется, чтобы в каждом новом поколении рождались особи, которые лучше, чем в предыдущем поколении.

Figure 1. схема алгоритма Генети́ческого

**Прекращение**: После того, как замена была произведена, используется критерий остановки, чтобы обеспечить основание для прекращения. Алгоритм завершится после достижения порогового значения пригодности. Он определит это решение как лучшее решение для населения.

**Преимущества генетического алгоритма**

* Обладает отличными параллельными возможностями.
* Он может оптимизировать различные задачи, такие как дискретные функции, многокритериальные задачи и непрерывные функции.
* Он дает ответы, которые со временем улучшаются.
* Генетическому алгоритму не нужна производная информация.

**Ограничения**

* Он неэффектив при решении простых задач.
* Отсутствие надлежащей реализации может привести к сходимости алгоритма к неоптимальному решению.
* Качество окончательного решения не гарантируется.
* Повторяющееся вычисление значений пригодности может привести к возникновению вычислительных проблем.
  1. **Задача распределения нагрузки и учебных групп (задача составления расписания занятий)**

Когда мы составляем расписание занятий, мы должны учитывать множество требований (количество профессоров, студентов, классов и аудиторий, размер классной комнаты, лабораторное оборудование в классе и многие другие). По степени важности эти требования можно разделить на несколько групп.

**Жесткие требования** (если вы нарушите одно из них, то расписание не выполнимо):

* Класс можно разместить только в свободном классе.
* Ни у одного профессора или студенческой группы не может быть более одного класса одновременно.
* В классе должно быть достаточно мест для размещения всех студентов.
* Чтобы поместить класс в класс, в классе должно быть лабораторное оборудование (в нашем случае компьютеры), если этого требует класс.

**Мягкие требования** (могут быть нарушены, но расписание все еще выполнимо):

* Предпочтительное время занятий профессорами.
* Предпочтительная аудитория профессоров.
* Распределение (во времени или пространстве) занятий по студенческим группам или профессорам.

Жесткие и мягкие требования, конечно, зависят от ситуации. В этом примере реализованы только жесткие требования. Начнем с объяснения объектов, составляющих расписание занятий.

**Данные задачи**

* Список профессоров
* Список групп
* Список классов
* Список курсов
* Список аудитории
  1. **Решение задачи на основе генетического алгоритма**
     1. ***Хромосома***

Первое, что мы должны учитывать, когда имеем дело с генетическим алгоритмом, - это то, как представить наше решение таким образом, чтобы оно было осуществимо для генетических операций, таких как кроссовер и мутация. Также мы должны знать, как определить, насколько хорошо наше решение. Другими словами, мы должны иметь возможность вычислить значение пригодности нашего решения.

Мы предполагаем, что в каждом классе максимум 4 пары в день, а рабочие дни с понедельника по пятницу

Мы можем использовать std :: vector размером 4 \* 5 \* number\_of\_rooms. Слот должен быть std :: list, потому что во время выполнения нашего алгоритма мы разрешаем несколько классов в одном временном и пространственном слоте. Существует дополнительная хеш-карта, которая используется для получения первого пространственно-временного слота, в котором класс начинается (его положение в векторе) с адреса объекта класса. Каждая пара класса имеет отдельную запись в векторе, но в хэш-карте есть только одна запись для каждого класса.

Хромосомы представлены классом Schedule, и он хранит представление расписания класса в этих двух атрибутах:

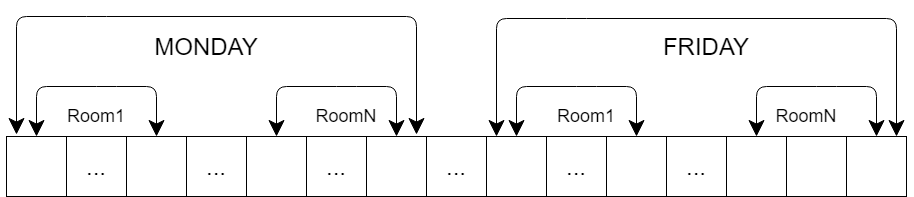
**Vector<list<courseclass\*>> \_slots;**

Figure 2. представление Хромосомы

**Hash\_map<courseclass\*, int> \_classes;**

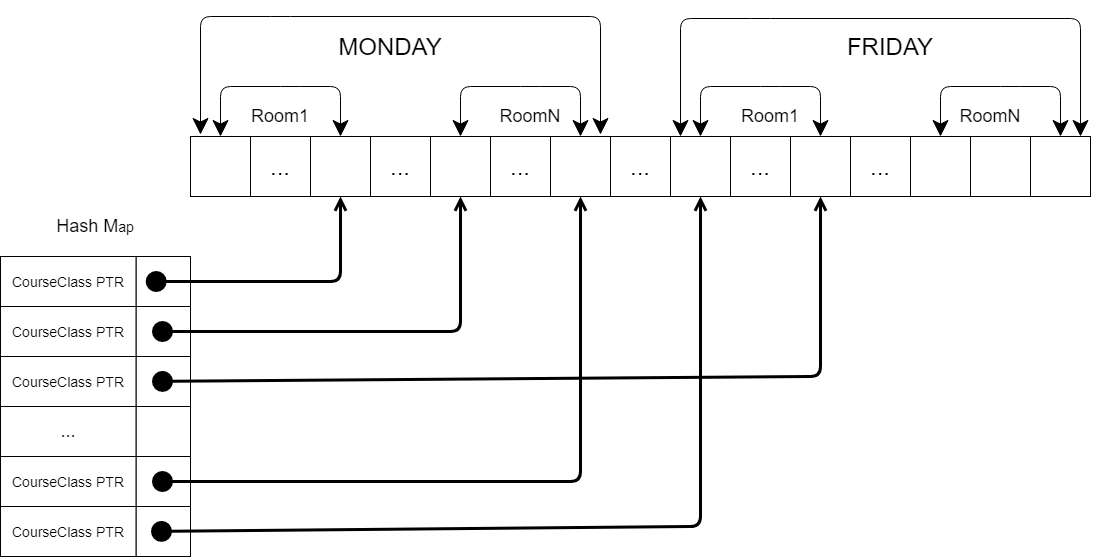
****

Figure 3. дополнительная хеш-карта

Кроме того, хромосома должна хранить свое значение приспособленности и параметры, которые используются в генетических операциях.

Фитнес-значение:

**Float \_fitness;**

Флаги соответствия требованиям класса:

**Vector<bool> \_criteria;**

Количество точек пересечения таблиц родительских классов

**Int \_numberofcrossoverpoints;**

Количество классов, которые случайным образом перемещаются при единственной операции мутации

**Int \_mutationsize;**

Вероятность того, что произойдет кроссовер:

**Int \_crossoverprobability;**

Вероятность появления мутации

**Int \_mutationprobability;**

* + 1. ***Функция фитнеса***

Теперь нам нужно присвоить хромосоме значение пригодности. Как мы уже говорили, для расчета соответствия расписанию занятий используются только жесткие требования. Вот как мы это делаем:

Каждый класс может иметь от 0 до 5 баллов.

* Если класс использует свободную комнату, мы увеличиваем его оценку.
* Если классу требуются компьютеры, и он находится в комнате с ними, или он не требует их, мы увеличиваем оценку класса.
* Если класс находится в комнате с достаточным количеством свободных мест, угадайте, что, мы увеличиваем его оценку.
* Если у профессора в это время нет других классов, мы снова увеличиваем оценку класса.
* Последнее, что мы проверяем, это то, есть ли в какой-либо из групп учащихся, которые посещают класс, какой-либо другой класс в то же время, и если они этого не делают, мы увеличиваем оценку класса.

Если класс нарушает правило в любом временном интервале, который он занимает, его оценка для этого правила не увеличивается.

Общий балл по расписанию занятий - это сумма баллов всех классов.

Значение пригодности рассчитывается как schedule\_score / maximum\_score, а maximum\_score равно number\_of\_classes \* 5.

Значения пригодности представлены числами с плавающей запятой одинарной точности (float) в диапазоне от 0 до 1.

* + 1. **Кроссовер**

Операция кроссовера объединяет данные в хэш-картах двух родителей, а затем создает вектор слотов в соответствии с содержимым новой хэш-карты. Кроссовер «разбивает» хэш-карты обоих родителей на части случайного размера. Количество частей определяется количеством точек кроссовера (плюс одна) в параметрах хромосомы. Затем он поочередно копирует части от родителей к новой хромосоме и формирует новый вектор слотов.

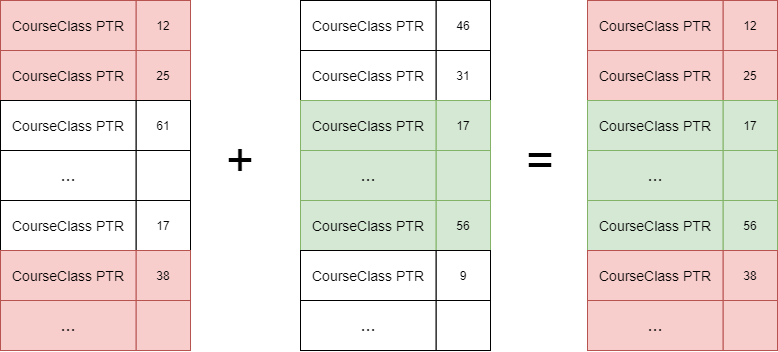


Figure 4. Операция кроссовера

* + 1. **Мутация**

Операция мутации очень проста. Он просто случайным образом берет класс и перемещает его в другой случайно выбранный слот. Количество классов, которые будут перемещены за одну операцию, определяется размером мутации в параметрах хромосомы.

* + 1. **Алгоритм**

Генетический алгоритм довольно прост. Для каждого поколения он выполняет две основные операции:

1. Случайным образом выбирает N пар родителей из текущей популяции и производит N новых хромосом, выполняя операцию кроссовера на паре родителей.

2. Случайным образом выбирает N хромосом из текущей популяции и заменяет их новыми. Алгоритм не выбирает хромосомы для замены, если они входят в число лучших хромосом в популяции.

И эти две операции повторяются до тех пор, пока лучшая хромосома не достигнет значения пригодности, равного 1 (что означает, что все классы в расписании соответствуют требованиям). Как упоминалось ранее, генетический алгоритм отслеживает M лучших хромосом в популяции и гарантирует, что они не будут заменены, пока они находятся среди лучших хромосом.

1. **Конструкторская часть**
   1. **Выбор используемых типов и структур данных**

Профессор: Класс «Профессор» имеет идентификатор и имя профессора. Он также содержит список классов, которые преподает профессор.

Студенческая группа: У класса studentsgroup есть идентификатор и название студенческой группы, а также количество студентов (размер группы). Он также содержит список классов, которые посещает группа.

Аудитории: Класс Room содержит идентификатор и название класса, а также количество мест и информацию об оборудовании (компьютерах). Если в классе есть компьютеры, предполагается, что на каждое место будет компьютер. Идентификаторы генерируются внутри и автоматически.

Курс: У класса Course есть идентификатор и название курса.

Класс: Courseclass содержит ссылку на курс, к которому относится класс, ссылку на преподавателя и список групп студентов, которые посещают класс. Он также хранит, сколько мест (сумма размеров студенческих групп) необходимо в классе, если классу требуются компьютеры в классе, и продолжительность занятия (в часах).

* 1. **Структура и состав классов**

В этом разделе будут рассмотрена структура и состав классов

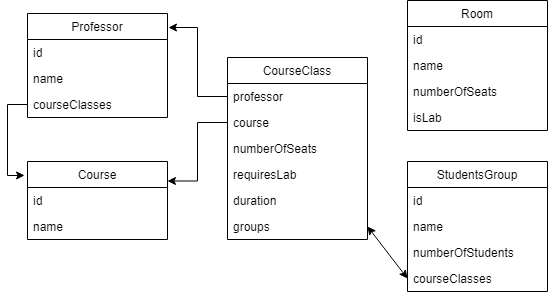
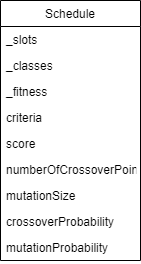


Figure 5. Структура и состав классов

1. **Технологическая часть**
   1. Выбор и обоснование языка программирования и среды разработки

В качестве языка программирования был выбран Python т.к.:

• Это язык Python четкой формой, четкой структурой и кратким  
синтаксисом.

• Данный язык программирования объектно-ориентирован.

В качестве среды разработки была выбрана «Visual Studio 2017» т.к. Oнa имеет множество удобств, которые облегчают процесс написания и отладки  
кода.s

* 1. **Сведения о модулях программы**

Gui.py – интерфейс

Algorithm.py - описание алгоритма

Course.py - описание курсов

Courseclass.py - описание классов

Professor.py - описание профессоров

Room.py - описание аудитории

Schedule.py - описание хромосомы

Studentsgroup.py - описание групп

Configuration.py - чтение данные из файла

* 1. **Интерфейс программы**

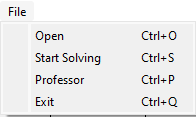


Figure 6. Интерфейс программы

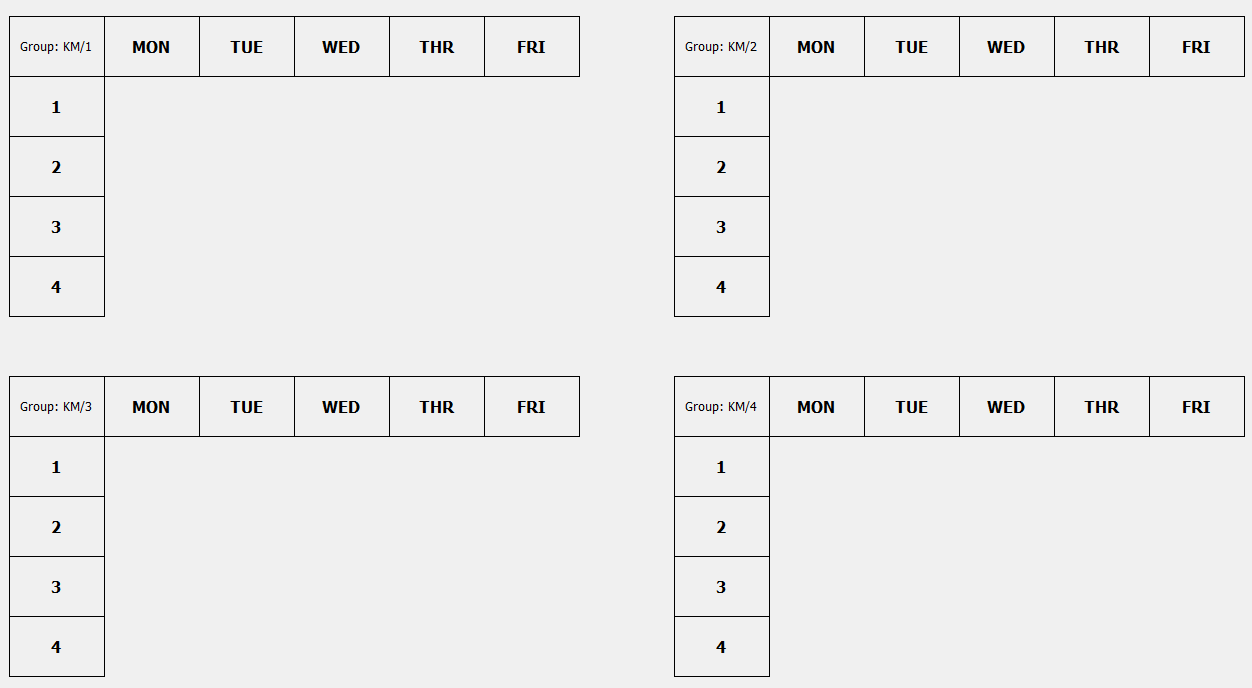


Figure 7. Интерфейс программы

1. **Экспериментальная часть**
   1. Цель эксперимента:

Целью эксперимента является проверка правильности выполнения  
поставленной задачи

**Результат по группам**

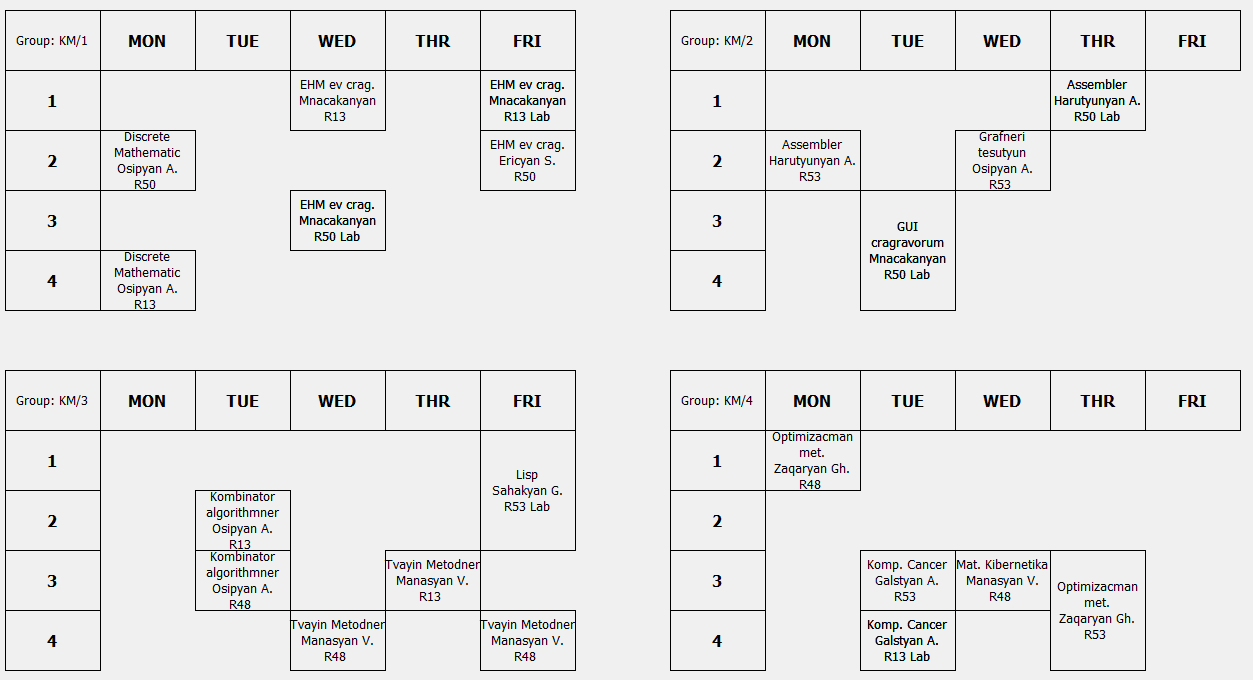


Figure 8. Результат по группам

**Результат по профессору**

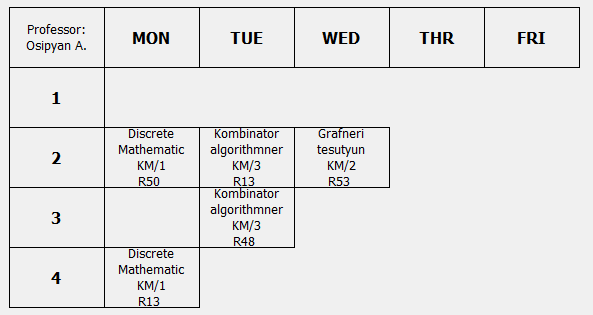


Figure 9. Результат по профессору

1. **Заключение**

В ходе практики реализован алгоритм Генети́ческого и разработано приложение распределения нагрузки и учебных групп по группам и по профессору

1. **Список использованной литературы**

1. Genetic Algorithm *[https://www.sciencedirect.com/topics/engineering/genetic-algorithm]*

*2. Using Genetic Algorithms to Schedule Timetables*

*[https://towardsdatascience.com/using-genetic-algorithms-to-schedule-timetables]*