МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ

по летней практике

Тема: Генетические алгоритмы

Студенты гр. 0382 Азаров М.С.
Шангичев В.А.
Санников В.А.
Диденко Д.В.

Преподаватель _____ Жангиров. Т.Р.

Санкт-Петербург
2022

Цель работы.

Ознакомится с понятием *Генетического алгоритма (ГА). Решить поставленную задачу с помощью приобретенных знаний в области методов ГА. Разработать полноценное приложение решающую данную задачу , с удобным пользовательским интерфейсом и визуализацией работы алгоритма.*

Задание.

Разработать и реализовать программу, решающую одну из оптимизационных задач (файл “Варианты”) с использованием генетических алгоритмов (ГА), а также визуализирующая работу алгоритма. В качестве языков программирования можно выбрать: C++, C#, Python, Java, Kotlin. Все параметры ГА необходимо определить самостоятельно исходя из выбранного варианта. Разрешается брать один и тот же вариант разным бригадам, но при условии, что будут использоваться разные языки программирования.

Вариант: 11

Минимальное остовное дерево (*minimal spanning tree — MST*) в связанном взвешенном неориентированном графе — это остовное дерево этого графа, имеющее минимальный возможный вес, где под весом дерева понимается сумма весов входящих в него рёбер.

Задача заключается в нахождении минимального остовного дерева для вводимого графа.

Входные данные:

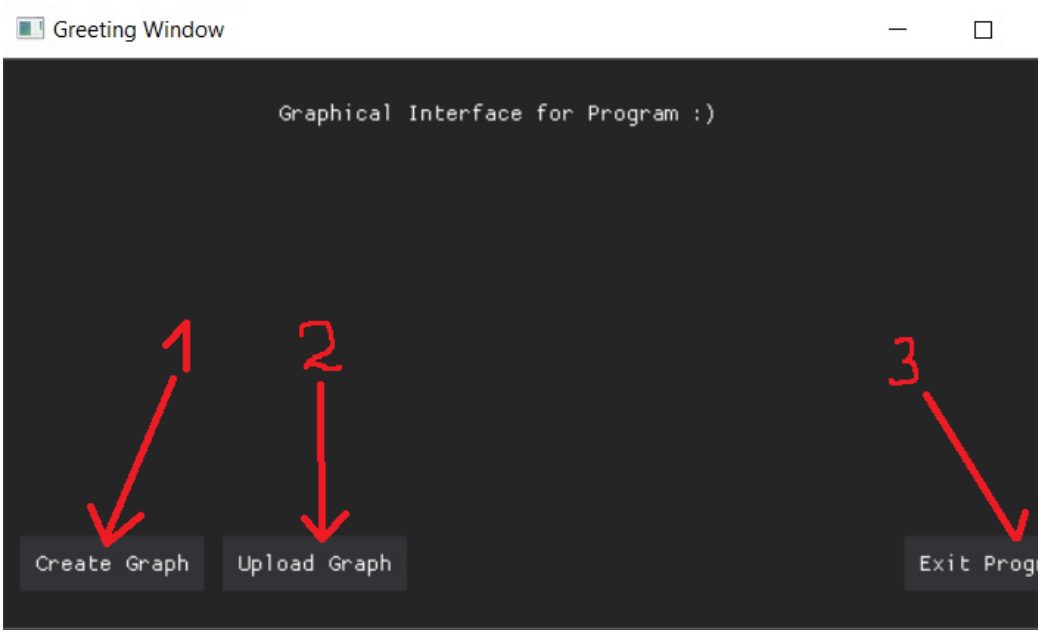
- Список вершин
- Список ребер

Язык: Python

РАЗДЕЛ 1. Формирование прототипа GUI и выбор метода решения задачи

1. Предварительный набросок интерфейса программы:

- Главное меню:



1 — Кнопка создания графа.

2 — Кнопка загрузки графа из файла.

3 — Кнопка выхода из программы.

- Основное окно программы.

Рисунок 2: Предварительный набросок окна "Основное окно

1 — Вернутся в главное меню.

2 — Досрочное завершение алгоритма.

3 — Кнопка отвечающая за запуск алгоритма.

4 — Область визуализации работы ГА.

5 — Область управлением визуализации алгоритма.

6 — Область настроек параметров алгоритма ГА.

2. Описание сценария взаимодействия пользователь с программой.

«Главное меню»

При запуске приложения открывается первое окно «Главное меню». В нем пользователь может выбрать следующие действия:

- «создать граф» - при выборе этого действия будет открываться дополнительное окно для ввода графа . После успешного ввода пользователю откроется «Основное окно программы».
- «загрузить граф» - открываться проводник с возможностью выбора нужного файла. (Формат файла еще но уточнен и требует дальнейшего рассмотрения) После выбора файла пользователю откроется «Основное окно программы».
- «выйти из программы» - завершение программы.

«Основное окно программы»

После того как введен или загружен нужный граф пользователю открывается «Основное окно программы».

В этом окне будут присутствовать область настроек параметров алгоритма (см. Рисунок 2, пункт 6), такие как размер популяции, вероятность кроссинговера и т. д. Эти параметры пользователю нужно будет указать (либо оставить значения по умолчанию) перед запуском алгоритма.

После указания параметров алгоритма пользователю нужно будет нажать кнопку «Запуска алгоритма» (см. Рисунок 2, пункт 3), после которой запустится демонстрация работы алгоритма, а область настройки параметров алгоритма перестанет быть доступной для изменений.

Демонстрация алгоритма будет показываться справа в области 4 на Рисунке 2. Каждая особь будет отображаться точкой на координатной плоскости, где по оси Y будет шкала весов каждой особи (вес особи — вес графа соответствующей этой особи), а по оси X поколения . Чтобы увидеть граф соответствующей особи, нужно будет нажать на нужную особь и ее граф откроется в доп. окне.

Это очень удобное представление работы алгоритма, так как чтобы увидеть лучшую особь поколения нужно посмотреть на самую нижнюю точку в одном

поколении. Родители, потомки (особи получившиеся в результате кроссинговера) и мутанты будут отображаться разными точками (будет легенда по которой можно понять кто есть кто).

Предполагается что визуализация алгоритма будет иметь два режима работы «Пошаговый» и «Автоматический». При пошаговом режиме каждый этап работы алгоритма и его последующая визуализация будет осуществляться по нажатию определенной клавиши. При автоматическом режиме пользователю не нужно будет нажимать клавиши, алгоритм будет переходить на следующий этап по истечению небольшого промежутка времени.

Управление режимами визуализации будет располагаться в области 5 Рисунок 2. Также в этой области будет находится кнопка «Перехода к концу алгоритма».

После схождения алгоритма откроется окно с окончательным ответом (минимальным остовным деревом). После нажатия кнопки «Ок» пользователь вернется на окно «Основное окно программы», в котором уже будут доступны для изменения параметры алгоритма. Таким образом пользователь сможет изменить параметры алгоритма для текущего загруженного графа или вернутся в «Главное меню» для указания нового графа.

Также во время выполнения алгоритма можно будет нажать кнопку «Досрочного завершения», в случае если алгоритм не сходится. В таком случае алгоритм прекратит работу, параметры алгоритма будут доступны для изменения и пользователь сможет указать новые параметры алгоритма, либо загрузить другой граф.

3. Определение и обоснование параметров, модификации ГА для решения выбранной задачи

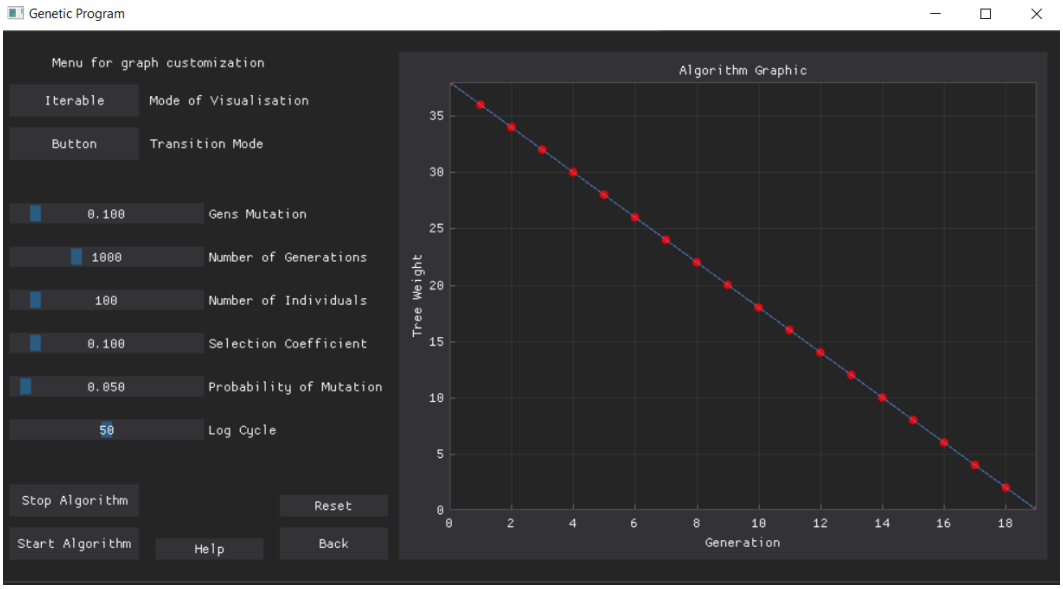
Для решения поставленной задачи будут проводиться эксперименты с различными вариантами кроссовера, отбора и мутации. Генетический код каждой особи представлен двоичным числом, количество разрядов в котором равно количеству ребер в графе. В качестве базового решения было решено использовать однородный кроссовер с отбором родителей по принципу рулетки

с последующей двоичной мутацией с вероятностью мутации особи около 5%, где каждый ген мутирует с вероятностью 10%, и элитарным отбором с долей элитных особей 10%. Численность популяции планируется содержать от 100 особей, причем будут уместны эксперименты с нефиксированным количеством особей. Такой выбор параметров обоснован необходимостью поддержания широкого генетического разнообразия в генерациях, т. к. в произвольном графе (в худшем случае в полном) большинство наборов ребер не будут являться остовными деревьями, и в случае скудного генетического разнообразия в некоторых случаях дерево получить будет невозможно.

Пробная версия фитнес-функции будет выдавать ε (очень малое положительное число), если генетический код особи не представляет дерево и 1 / <вес остовного дерева> в противном случае. Данная функция будет по мере необходимости дополняться. Например, интересна и достойна рассмотрения идея регуляции количества ребер. В таком случае необходимо будет определить, какое количество ребер у особей ведет к наиболее частому появлению генетических кодов, представляющих остовные деревья, и обеспечить сходимость к данному количеству ребер с помощью фитнес функции.

РАЗДЕЛ 2. Начало реализации

1. Начало реализации GUI:



В ходе данной итерации на первом окне «Главное» была реализована кнопка выхода из программы: при нажатии на кнопку "Exit Program" программа завершается и все окна закрываются. Это окно почти не претерпело визуальных изменений с этапа наброска в предыдущем разделе.

Также была реализована кнопка "Create Graph", которая отвечает за создание графа и переход на новое окно кастомизации алгоритма (на данный момент реализован лишь переход на «Основное окно алгоритма»).

В «Основное окно алгоритма» были реализованы следующие элементы интерфейса:

- Кнопка "Iterable/To End" - выбор режима визуализации алгоритма (итеративный или сразу в конец).
- Кнопка Button/SlideShow - переход при итерации по кнопке или слайдшоу режим (если выбран итеративный режим визуализации)
- Реализованы ползунки с параметрами алгоритма (при перемещении ползунка изменяются параметры)
- Reset - сбрасывает параметры на дефолтное состояние
- Back - возврат в меню создания графа(при этом настроенные параметры алгоритма буду сброшены)
- Help - окно помощи для пользователя, где будут описаны все параметры алгоритма и какие кнопки за что отвечают (при переходе на данное окно и по возвращении обратно, параметры ползунков, которые настроил пользователь, сохранятся)
- Stop algorithm - останавливает алгоритм
- Start algorithm - начинает выполнение алгоритма, собирая информацию о параметрах по ползункам и по двум кнопкам.

2. Начало реализации алгоритма:

Реализация ГА.

Генетический код каждой особи представлен двоичным числом, количество разрядов в котором равно количеству ребер в графе. В настоящий момент в проекте используется следующая фитнес-функция:

```
fitness_func = (1/tree_cost) - (2*epsilon*|num_edges - n + 1|)/(|n^2 - 3*n + 2|)
```

, где `tree_cost` – вес минимального остовного дерева, если генетический код представляет дерево, и сумма всех ребер в графе, если генетический код представляет произвольный подграф.

`epsilon` – 1 / сумма всех ребер в графе.

`num_edges` - количество ребер в графе.

`n` – количество вершин в графе.

Данная функция поощряет создание деревьев в графе, а также поддерживает количество ребер в подграфах числом, равным `n - 1`.

Оптимальное количество ребер у особей для получения наибольшего количество деревьев на этапе кроссовера пока что не было математически установлено, однако экспериментально было выявлено превосходство этой версии над базовым решением (`1/tree_cost`).

Следующее изменение по сравнению с базовой версией (однородный кроссовер с отбором родителей по принципу рулетки с последующей двоичной мутацией с вероятностью мутации особи около 5%, где каждый ген мутирует с вероятностью 10%, и элитарным отбором с долей элитных особей 10%.) коснулось отбора особей в кроссовер. Вместо рулетки было решено использовать *панмиксию*, потому что, как и было замечено в предыдущем отчете, в данной задаче очень важно поддерживать широкое генетическое разнообразие, которого рулеточный тип отбора не был способен обеспечивать.

Численность популяции в текущей версии составляет 100 особей, причем улучшений работы алгоритма при увеличении данного числа не наблюдается.

Архитектура кода.

Архитектурной основой является класс `GA`, расположенный в файле `ga.py`. Данный класс содержит все параметры, необходимые для работы генетического алгоритма (указатели на функции репродукции, мутации, кроссовера и все соответствующие для их работы параметры). Класс содержит метод `run` и `run_by_step`. Первый метод запускает генетический алгоритм и предназначен для тестирования корректности ГА и проведения экспериментов для поиска наиболее оптимальных параметров. А второй реализует такой же механизм, но реализован в виде генератора, возвращающего на каждой генерации словарь, характеризующий изменение популяции на протяжении одной итерации генетического алгоритма (репродукция, мутация, отбор). Второй метод

предназначен для непосредственного более удобного применения алгоритма в разработке GUI.

Особь представлена классом `Individual`, расположенного в файле `individual.py`. Класс содержит поля генетического кода, значения фитнес-функции, веса графа, представленного генетическим кодом (будет использоваться позже) и два метода для удобства представления.

Помимо файлов, перечисленных выше, в проекте содержатся файлы:

- `mutations_functions.py`
- `reproduction_functions.py`
- `selection_functions.py`
- `fitness_functions.py`

содержащие различные реализации функций мутации, репродукции, селекции и фитнес-функций.

Файл `graph_functions.py` содержит функции для работы с графами, а файл `ga_functions.py` – содержит вспомогательные для ГА функции, например, функция логирования и функция создания новых особей. В файле `example_work_GA.py` размещен пример успешной работы алгоритма для полного графа на 17 вершинах. А в файле `example_for_GUI.py` продемонстрировано как разработчикам GUI пользоваться генератором `ga. run_by_step()`.

РАЗДЕЛ 3. Работающий прототип

В ходе данной итерации был доработан функционал GUI, взаимодействие GUI с ГА, доработан ГА, реализовано логирование в файл.

1. GUI:

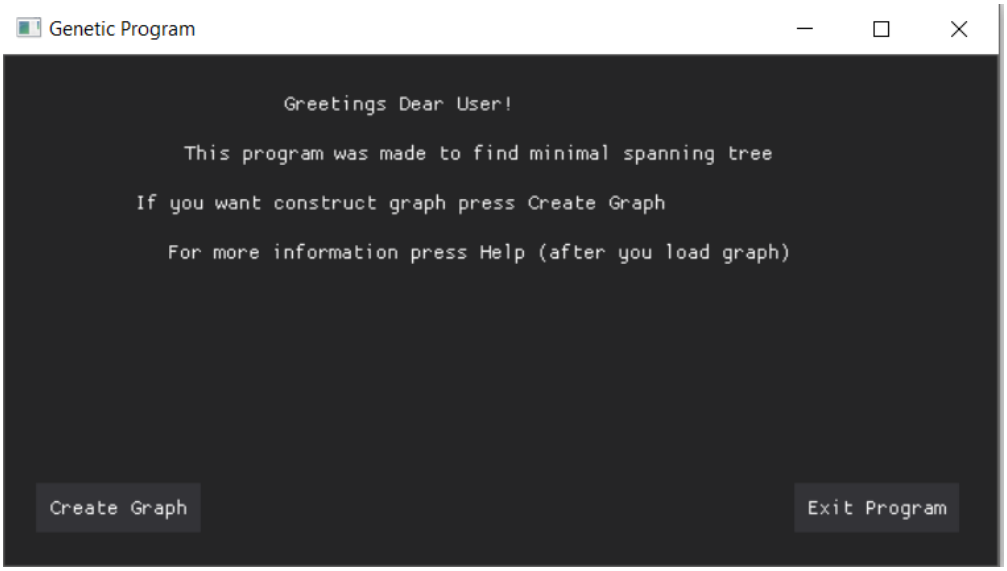


Рисунок 1 - Начальное окно

Как видно из рисунка 1, данное окно не претерпело значительных изменений, за исключением того, что добавлена мини-инструкция по середине окна, убрана кнопка Upload Graph (она перенесена на следующее окно) и запрещено изменение размера окна пользователем.

По нажатии кнопки Create Graph пользователь переходит на окно создания графа. См. Рисунок 2.

Рисунок 2 – Окно создания графа

Как видно на рисунке, окно состоит из 5-ти блоков, каждый из которых отвечает за определенный задачи. В данном окне уже программно полностью реализован весь функционал ввода графа.

Блок 1: с помощью строк Vertex1, Vertex2 и Weight of Edge можно соединить две вершины ребром (вводится номер первой вершины, второй и вес ребра между ними), по кнопке Add граф рисуется в окне 5. Если попытаться соединить одни и те же вершины, то программа никак не реагирует на запрос пользователя, однако если поменять вес ребра, то программа нарисует граф с новым весом ребра. Если указанной вершины нет , то она создастся в графе.

Блок 2: в строку Vertex to Delete вводится номер вершины, которую надо удалить, по кнопке Delete Vertex вершина удаляется и в окне 5 рисуется новый граф. Если попытаться удалить одну и ту же вершину, то программа никак не отреагирует на запрос пользователя.

Блок 3: с помощью строк Vertex1 и Vertex2 можно удалить ребро между вершинами (вводится номер первой вершины и второй), по кнопке Delete Edge

ребро удаляется и рисуется новый граф. Если попытаться удалить не существующее ребро, то программа никак не отреагирует на запрос пользователя.

Блок 4: кнопка *Finish* завершает создание графа, который передается в виде списка смежности следующим образом: графом является список списков. В списке с индексом i хранятся все ребра, инцидентные вершине i в виде (вес, <номер вершины, инцидентной вершине i>). Кнопка *Clear* очищает граф. Кнопка *Back* возвращает пользователя на начальное окно, при этом граф очищается. С помощью кнопки *Upload Graph* можно загрузить граф файлом типа .txt, если файл содержит некорректное представление графа, то программа выдает ошибку (граф представлен в виде списка смежности по строкам: вершина, вершина, вес ребра). Кнопка *Save* сохраняет граф файлом типа .txt или изображением.

Блок 5: блок отображения графа, в данном блоке появляется граф по введенным пользователем данным.

Примеры сообщений об ошибках см. на Рисунках 3-5.

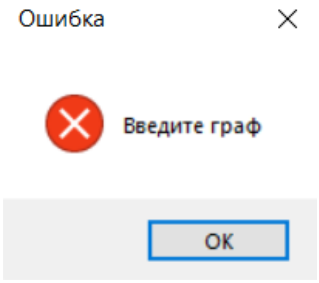


Рисунок 3 – Ошибка при нажатии кнопки Finish с пустым графом.

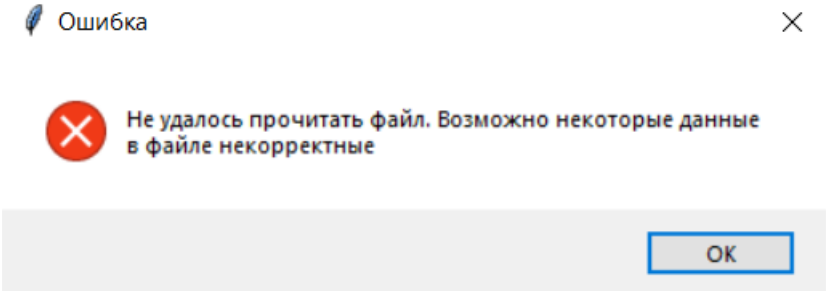


Рисунок 4 – Ошибка при нажатии кнопки Upload Graph и выбора некорректного файла.

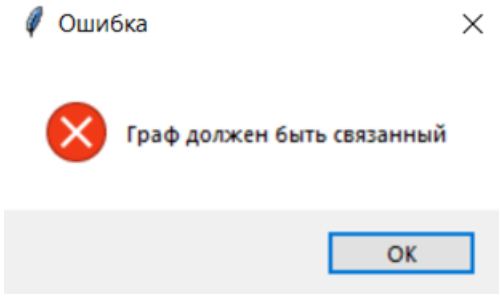


Рисунок 5 – Ошибка при вводе несвязного графа.

Пример построения графа см. на Рисунке 6

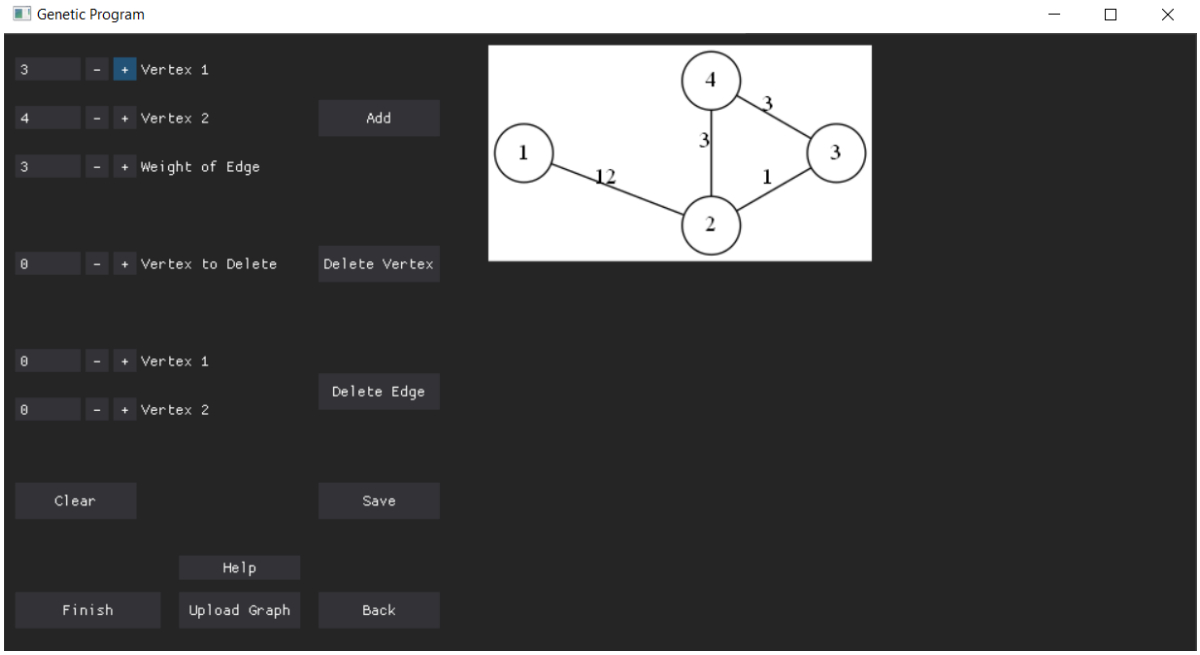


Рисунок 6 – Пример построения графа.

Как только граф создан и нет никаких ошибок, то пользователь переходит на окно алгоритма см. Рисунок 7.

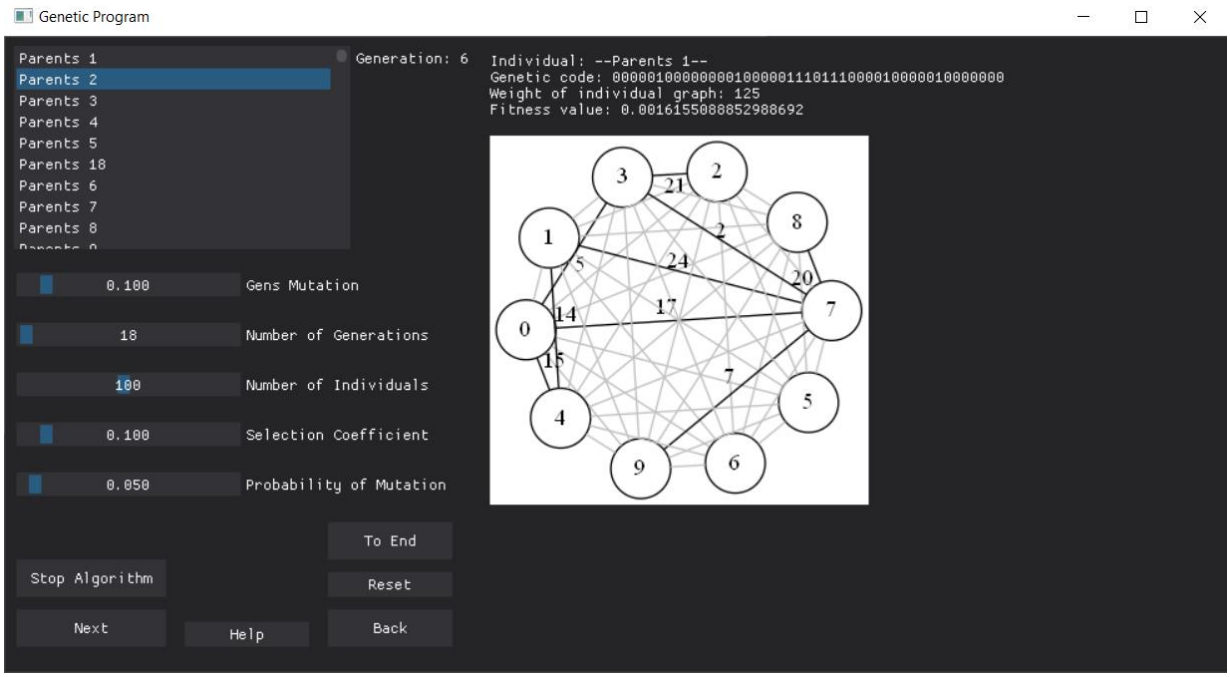


Рисунок 7 – Окно алгоритма при нажатии кнопки Start Algorithm

Как видно из данного рисунка окно претерпело серьезные изменения. Ввиду того, что точки на графике нельзя добавлять/удалять в runtime (библиотека не позволяла удалять/заменять/добавлять точки на графике и выдавала ошибку по памяти), было изменено отображение потомков, родителей и мутантов. Был реализован список в левом верхнем углу, который содержит потомков, родителей и мутантов на i-м шаге алгоритма. На любого из них можно нажать левой кнопкой мыши и граф данной особи отобразиться в правой части окна программы (у графа серые ребра – которые удалены, черные с весом – которые остались). На каждой итерации наверху списка стоит лучшее решение на данной итерации (можно проверить по фитнесс-функции). Над графом отображается номер поколения, особь, генетический код особи, вес графа и значение фитнесс-функции для определенной особи. Чтобы запустить алгоритм надо нажать кнопку Start Algorithm, которая заменяется на кнопку Next. Перед этим можно поменять параметры алгоритма (ползунки: Gens Mutation, Number of Generations, Number of Individuals, Selection Coefficient, Probability of Mutation). При запущенном алгоритме изменять параметры ползунков нельзя, так же как и сбрасывать их. Кнопка Stop Algorithm останавливает работу алгоритма, тогда можно поменять параметры (или сбросить их) и запустить программу с новыми

данными. Кнопка Next переходит на следующую итерацию алгоритма. Кнопка To End переходит на последнюю итерацию алгоритма, после нажатия на кнопку алгоритм начинает работу и показывается соответствующее сообщение. Когда алгоритм доходит до последней итерации или была нажата кнопка перехода в конец, то отображается лучшая особь с ее данными зеленым цветом (решение задачи) и открывается возможность редактирования параметров алгоритма и запуска программы с новыми параметрами, но с тем же графом. Кнопка Next заменяется обратно на Start Algorithm.

Пример вывода лучшего решения см. на Рисунке 8. Пример работы алгоритма, если нажата кнопка To End см. на Рисунке 9.

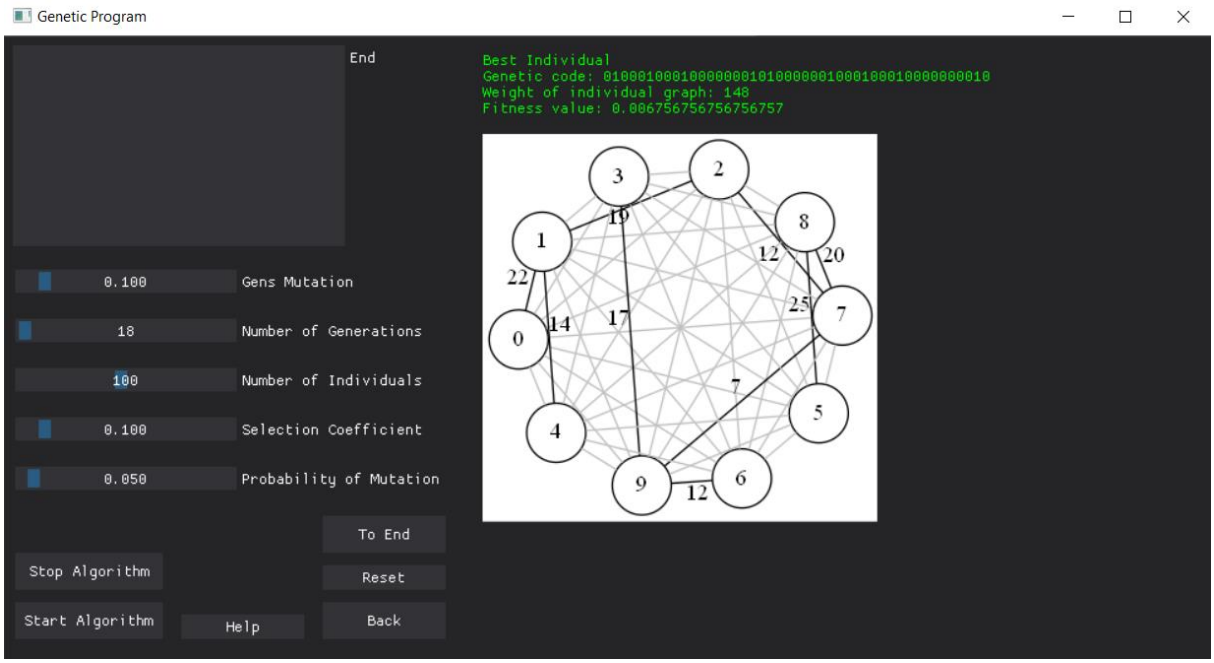


Рисунок 8 – Программа нашла лучшее решение

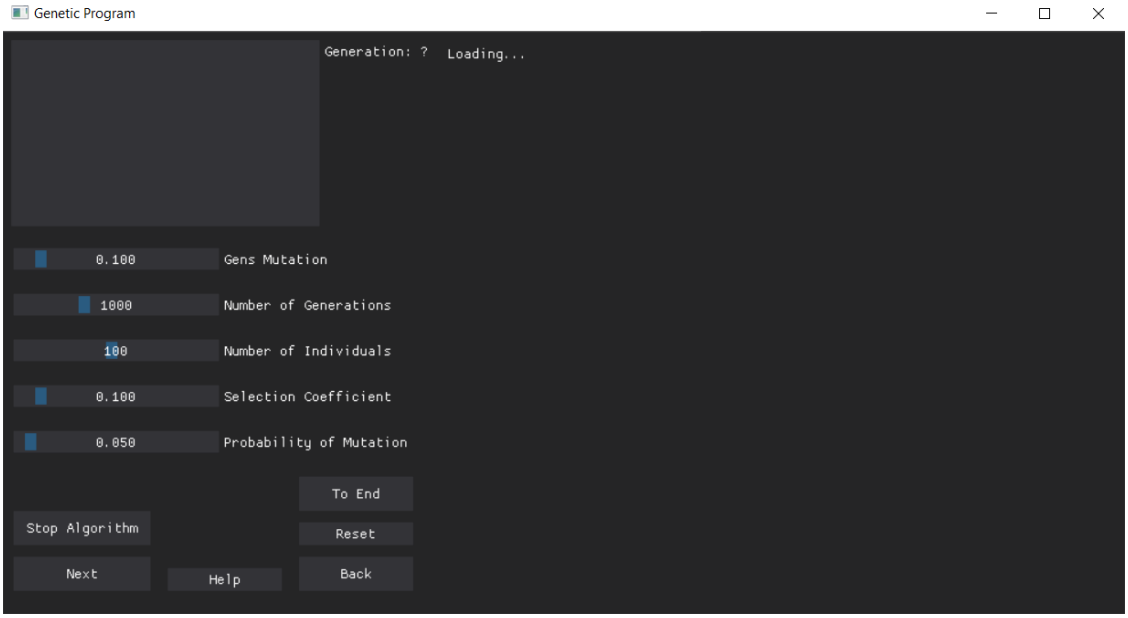


Рисунок 9 – Программа ищет лучшее решение

Для окна создания графа и окна алгоритма были добавлены окна помощи, которые описывают некоторый интерфейс на окне и подсказывают как взаимодействовать с данным окном. Примеры окон помощи см. на Рисунках 10 - 11.

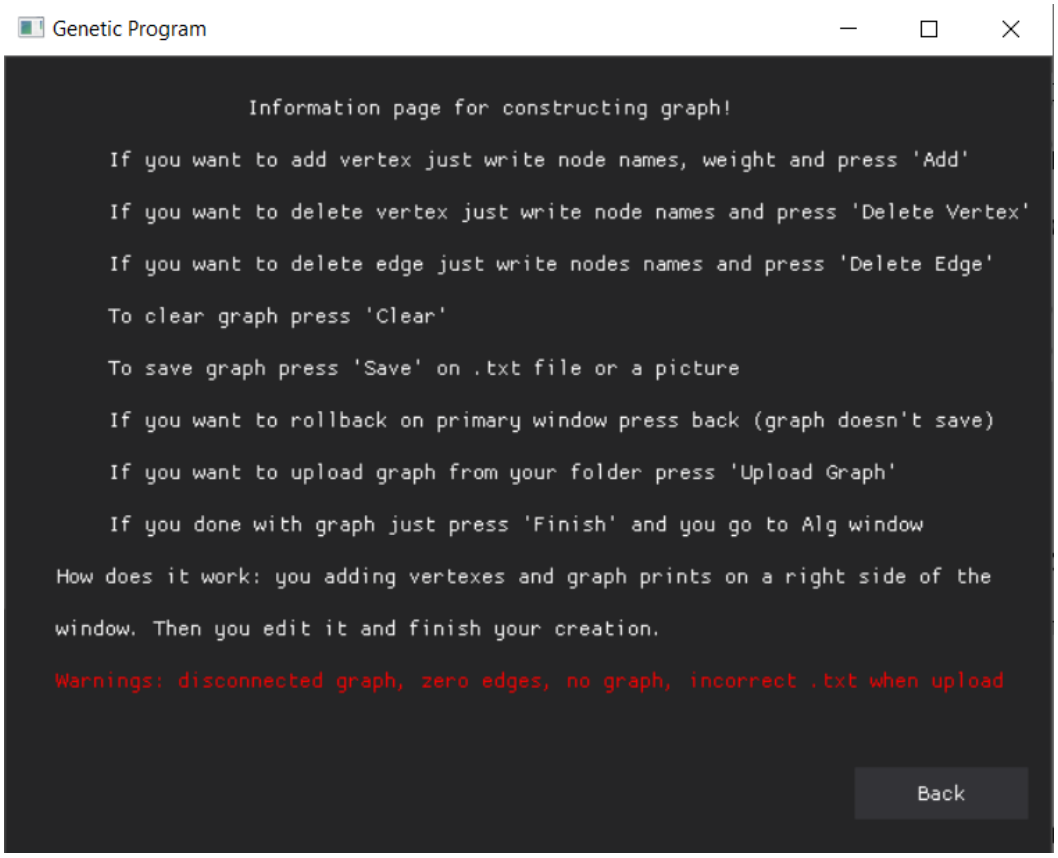


Рисунок 10 – Окно помощи для окна создания графа

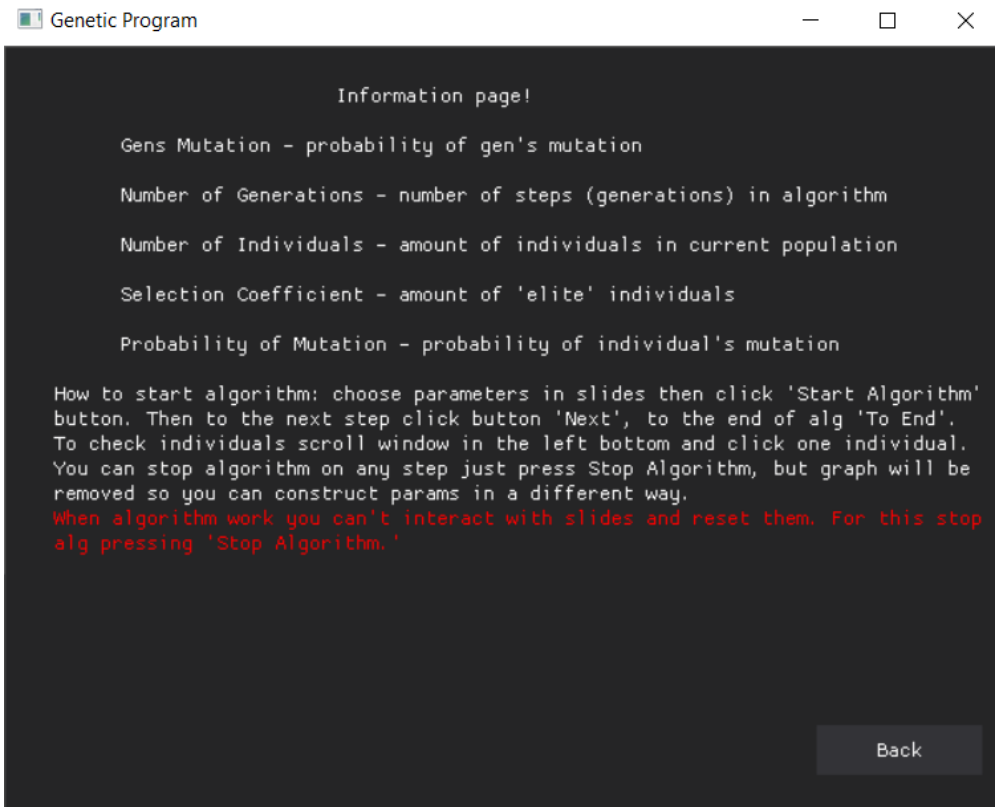


Рисунок 11 – Окно помощи для окна создания графа

2. Генетический алгоритм: в алгоритме было исправлено появление отрицательной фитнесс-функции, алгоритм был модифицирован и теперь сходится для полного графа на 20-ти вершинах

РАЗДЕЛ 4. Финальная версия

В ходе данной итерации программа не претерпела глобальных изменений в GUI и ГА. Однако были исправлены некоторые ошибки в работе обеих состовляющих.

1) Исправлен баг с черным окном при нажатии на кнопку Help в окне работы алгоритма при непосредственном запуске алгоритма.
2) Доработан ГА, теперь он гарантированно находит решение.
3) Доработано исключение, связанное с вводом графа с нулевым весом ребра.
4) Исправлен баг с игнорированием ошибки при создании графа, теперь кнопки не активны и нельзя вводить данные, пока не закроется окно ошибки.
5) Обработан тривиальный случай графа на двух вершинах.
6) Незначительные изменения в отображении графа и интерфейсе.