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

ОТЧЕТ

по учебной практике

Тема: Генетические алгоритмы

Студент гр. 2304 _____ Деменев К.О.

Студентка гр. 2304 _____ Иванова М.А.

Студент гр. 2304 _____ Шумилов А.В.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург
2024

Цель работы.

Изучить генетические алгоритмы, научиться применять их на практике. Разработать генетический алгоритм и GUI. Создать частично работающую программу, решающую задачу о неограниченном рюкзаке с помощью генетического алгоритма.

Задание. Вариант 1.

Задача о рюкзаке (1 рюкзак)

Дано N вещей, каждая i-я имеет вес W_i и стоимость C_i. Необходимо заполнить рюкзак с максимальной вместимостью по весу W_{max} вещами так, чтобы суммарная стоимость вещей в рюкзаке была максимальной. Можно класть несколько копий одной вещи в рюкзак.

Распределение ролей в команде.

- Деменев К.О. – разработка и реализация GUI;
- Иванова М.А. – написание отчета, частичная реализация алгоритма;
- Шумилов А.В. – организация работы в команде, разработка структуры проекта, частичная реализация алгоритма.

Генетический алгоритм был разработан совместно.

Выполнение работы.

Генетический алгоритм

Способы представления генома, отбора родителей и особей в следующее поколение, скрещивания и мутации, а также работа генетического алгоритма были описаны в предыдущем отчете.

К текущей итерации была изменена система *штрафов*. Таким образом, если суммарный вес вещей не превышает максимально допустимого, то функция приспособленности равняется суммарной стоимостей вещей. В случае перевеса суммарная стоимость вещей помножается на коэффициент, который вычисляется как $1 - \frac{\text{перевес}}{\max \text{допустимый вес}}$. Таким образом, значения с большим перевесом штрафуются сильнее, но не отбрасываются, как это было раньше.

За *вероятность мутации одного гена* в геноме в плотности мутации берется значение вероятности мутации целого генома. Новое значение изменяется на величину σ, равную $2\sum_{i=1}^{m} a(i)2^{-i}$, где m – параметр, а $a(i)$ – случайная величина, равная 1 с вероятностью $\frac{1}{m}$ или 0. Таким образом, новое значение гена = старое значение гена $\pm\,\sigma$, где знак + или - выбирается с равной вероятностью.

Организация кода

Для реализации генетического алгоритма были написаны следующие классы:

- Класс *Item* отвечает за предмет, помещаемый в рюкзак. Поля класса – цена (*cost*) и вес (*weight*) предмета. Методы класса представлены в табл. 1.
- Класс *Backpack* представляет собой особь поколения, т.е. одно из возможных решений задачи. Поля класса – геном (*genome*), представляющий решение задачи; вес (*weight*) и стоимость (*cost*) рюкзака. Методы класса представлены в табл. 2.

- Класс *Generation* представляет собой поколение. Поле класса - массив особей (*backpacks*), максимальное (*maxFitness*) и среднее (*averageFitness*) значения функции приспособленности по поколению. Методы класса представлены в табл. 3.
- Класс *AlgorithmParameters* содержит параметры генетического алгоритма: максимальный допустимый вес рюкзака (*maxBackpackWeight*), вероятность скрещивания (*crossingProbability*), вероятность мутации (*mutationProbability*), количество особей в поколении (*amountOfIndividsPerGeneration*), количество поколений (*maxAmountOfGenerations*). У класса нет методов.
- Класс *IterationInfo* содержит информацию о текущем решении: три лучших рюкзака (*bestBackpacks*), максимальное (*currentMaxFitness*) и среднее (*currentaverageFitness)* значения функции приспособленности по поколению. У класса нет методов.
- Класс *GeneticAlgorithm* является реализацией генетического алгоритма. Поля класса – массив вещей, введенных пользователем (*items*), параметры работы алгоритма (*parameters*). Методы класса представлены в табл. 4.

Таблица 1 – Использованные в классе *Item* методы

№ п/п	Метод	Входные аргументы	Возвращаемое значение	Предназначение
Инициализатор и перегруженные стандартные методы				
1.	__init__	cost: int, weight: int	-	Инициализатор
2.	__str__	-	str	Перегрузка вывода

Таблица 2 – Использованные в классе *Backpack* методы

№ п/п	Метод	Входные аргументы	Возвращаемое значение	Предназначение
Инициализатор и перегруженные стандартные методы				
1.	__init__	amountOfEachItems: list[int]	-	Инициализатор

2.	__str__	-	str	Перегрузка вывода
3.	__iter__	-	Iterator	Перегрузка итератора
4.	__le__ __lt__ __ge__ __gt__	other: 'Backpack'	bool	Перегрузка операторов сравнения
Методы, не являющиеся стандартными				
5.	calculateWeight	items: list[Item]	-	Расчет суммарного веса рюкзака
6.	calculateFitness	limitWeight: int, items: list[Item]	-	Расчет функции приспособленности рюкзака

Таблица 3 – Использованные в классе *Generation* методы

№ п/п	Метод	Входные аргументы	Возвращаемое значение	Предназначение
Инициализатор и перегруженные стандартные методы				
1.	__init__	backpacks: list[Backpack]	-	Инициализатор
2.	__str__	-	str	Перегрузка вывода
3.	__iter__	-	Iterator	Перегрузка итератора
4.	__len__	-	int	Перегрузка оператора длины
5.	__getitem__	key: int	Backpack	Перегрузка получения элемента по индексу
6.	append	item: Backpack	-	Перегрузка добавления элемента в конец массива
7.	expend	other: 'Generation'	-	Перегрузка добавления элементов
8.	remove	item: Backpack	-	Перегрузка удаления элемента по значению
Методы, не являющиеся стандартными				
9.	getBestBackpacks	-	list[Backpack]	Возвращает список из трех наиболее приспособленных рюкзаков в поколении
10.	getAverageFitness	-	float	Возвращает среднее значение функции приспособленности по поколению

11.	getMaxFitness	-	int	Возвращает максимальное значение функции приспособленности по поколению
12.	calculateWeight	items: list[Item]	-	Пересчитывает вес каждой особи поколения
13.	calculateFitness	limitWeight: int, items: list[Item]	-	Пересчитывает функцию приспособленности каждой особи поколения

Таблица 4 – Использованные в классе *GeneticAlgorithm* методы

№ п/п	Метод	Входные аргументы	Возвращаемое значение	Предназначение
Инициализатор и перегруженные стандартные методы				
1.	__init__	items: list[Item], parameters: AlgorithmParameters	-	Инициализатор
Методы, не являющиеся стандартными				
2.	generateRandomBackpack	-	Backpack	Генерирует рюкзак случайным образом
3.	generateRandomGeneration	-	Generation	Генерирует первое поколение случайным образом
4.	tournamentParentsSelection	generation: Generation	list[Backpack]	Осуществляет отбор N родителей турниром для скрещивания, N – объем популяции
5.	uniformCrossingForTwoParents	parents: tuple[Backpack, Backpack]	list[Backpack]	Проводит турнирный отбор среду двух родителей
6.	uniformParentsCrossing	selectedParents: list[Backpack]	list[Backpack]	Осуществляет равномерное скрещивание родителей, пока не будет получено N детей N – объем популяции
7.	densityMutationOneChild	child: Backpack	-	Осуществляет мутацию одного генома

8.	densityChildrenMutation	children: list[Backpack]	-	Осуществляет мутацию популяции
9.	eliteChoice	selectedParents: list[Backpack], producedChildren: list[Backpack]	Generation	Производит элитарный отбор особей в следующее поколение
10.	drawPlot	maxFitness: list[int], averageFitness: list[float]	-	Строит график функции приспособленности
11.	outputGenerationInfo	generation: Generation, generationNumber: int	-	Выводит информацию о поколении
12.	outputBackpacks	backpacks: list[Backpack]	-	Выводит информацию о массиве рюкзаков
13.	getSolution	-	list[IterationInfo]	Решает задачу о неограниченном рюкзаке с помощью генетического алгортима

Реализация графического интерфейса

Для написания GUI был использован Qt-designer.

Изменена структура интерфейса: теперь все приложение занимает одно окно, которое делится на вкладки: «инструкция», «итерация алгоритма», «история операций», «результат».

Теперь при запуске приложения в полях параметров алгоритма сразу отображаются значения по умолчанию. В случае, если параметры работы алгоритма или данные не будут введены, кнопка запуска алгоритма будет неактивна.

Ввод данных вручную теперь выглядит как заполнение таблицы, при вводе данных из файла область видимости ограничена расширением .txt. Добавлена возможность случайной генерации данных и перезапуска алгоритма.

При выводе информации о текущей итерации выводится информация о трех наиболее приспособленных геномах в поколении: таблица, отражающая заполненность каждого рюкзака, их стоимоть, вес и оставшееся для

заполнения место, а также отображается график со средним и максимальным значениями функции приспособленности.

Новая структура GUI отражена на рисунках 1-11.

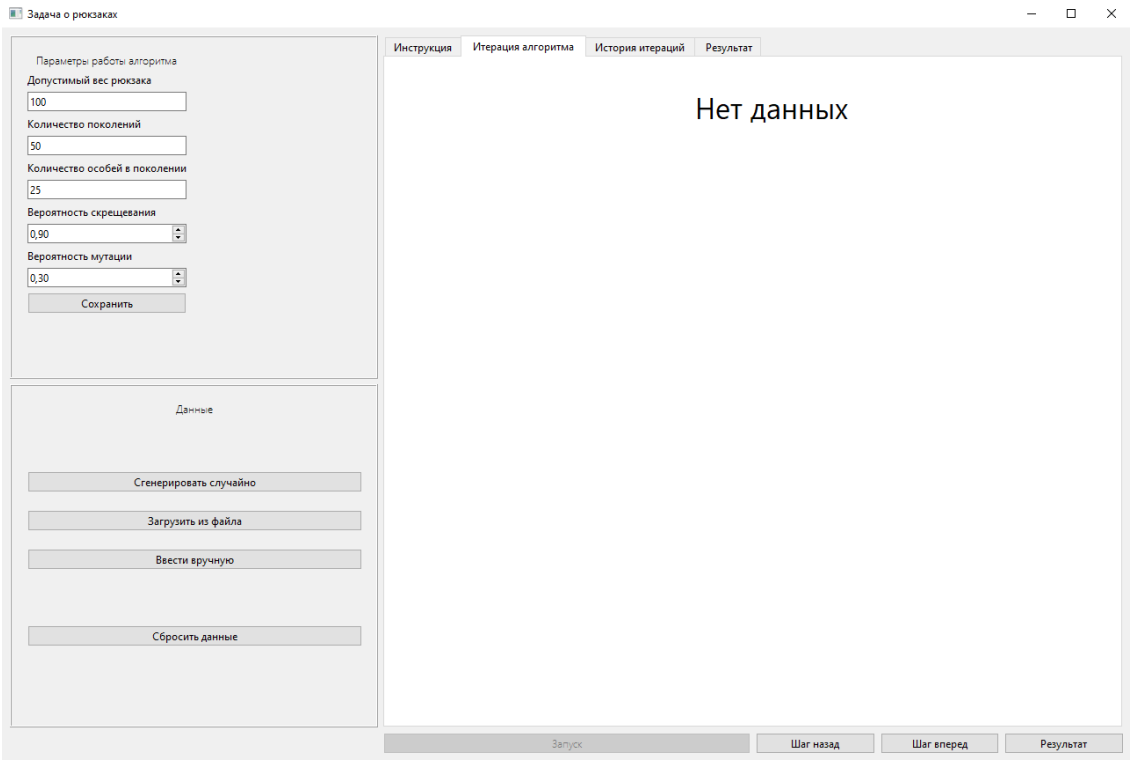

Рисунок 1 - Окно при запуске программы

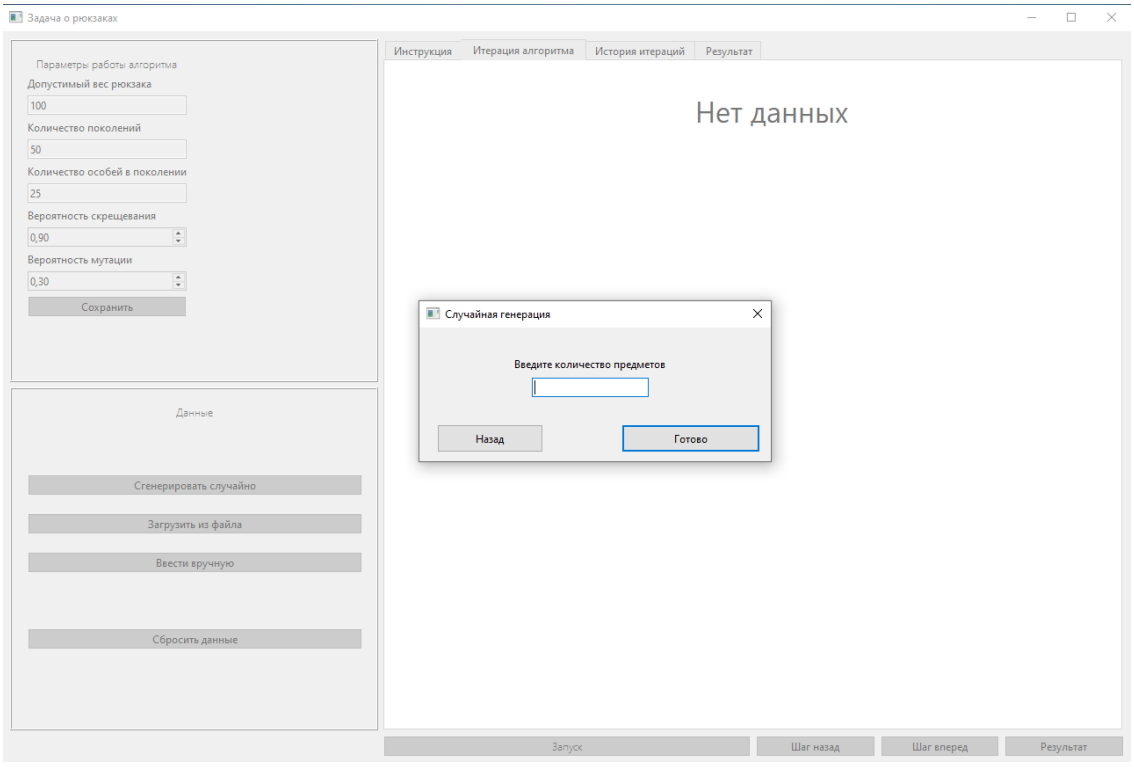

Рисунок 2 - Окно при выборе случайной генерации данных

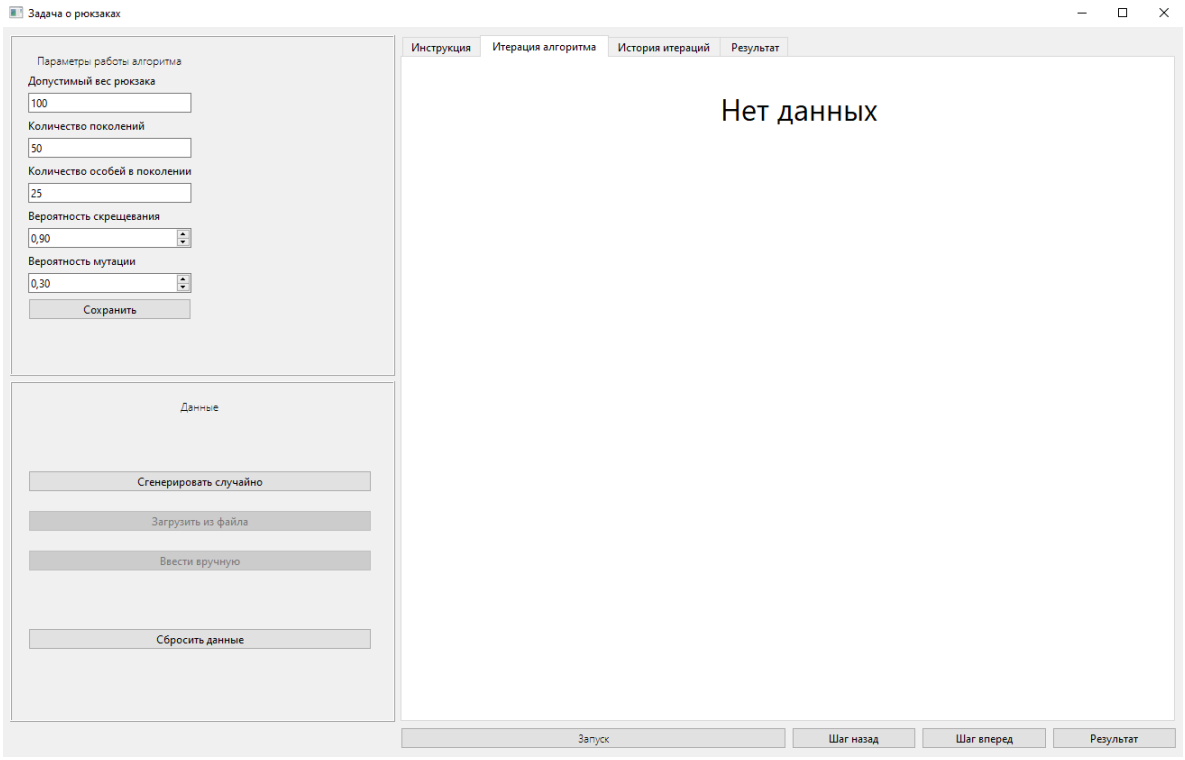

Рисунок 3 - Окно после выбора ввода данных случайной генерацией

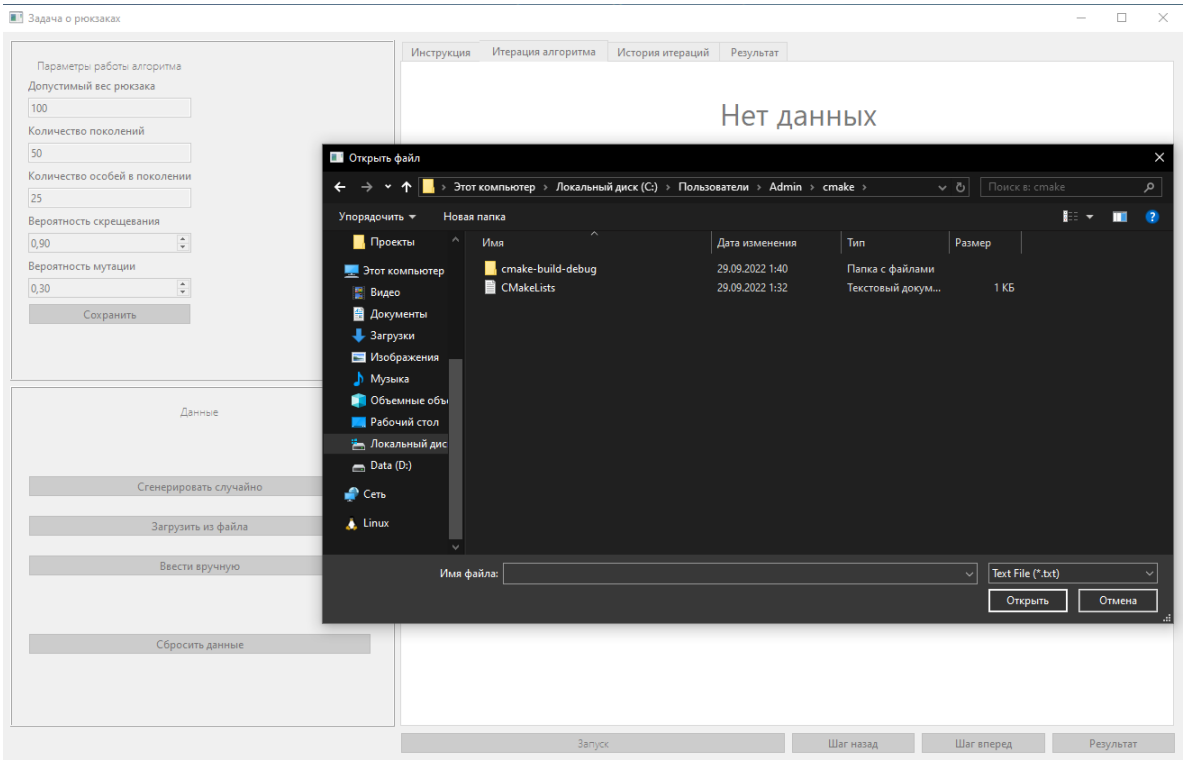

Рисунок 4 - Окно при выборе ввода данных из файла

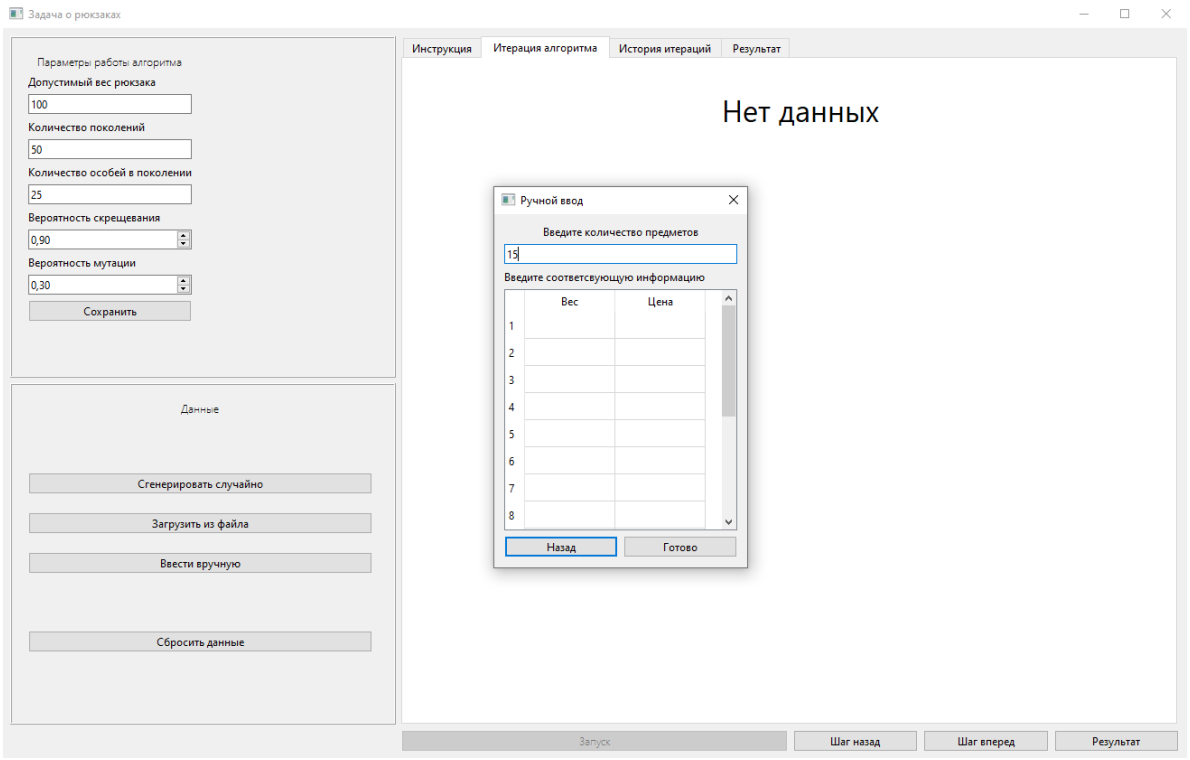

Рисунок 5 - Окно при выборе ввода данных вручную

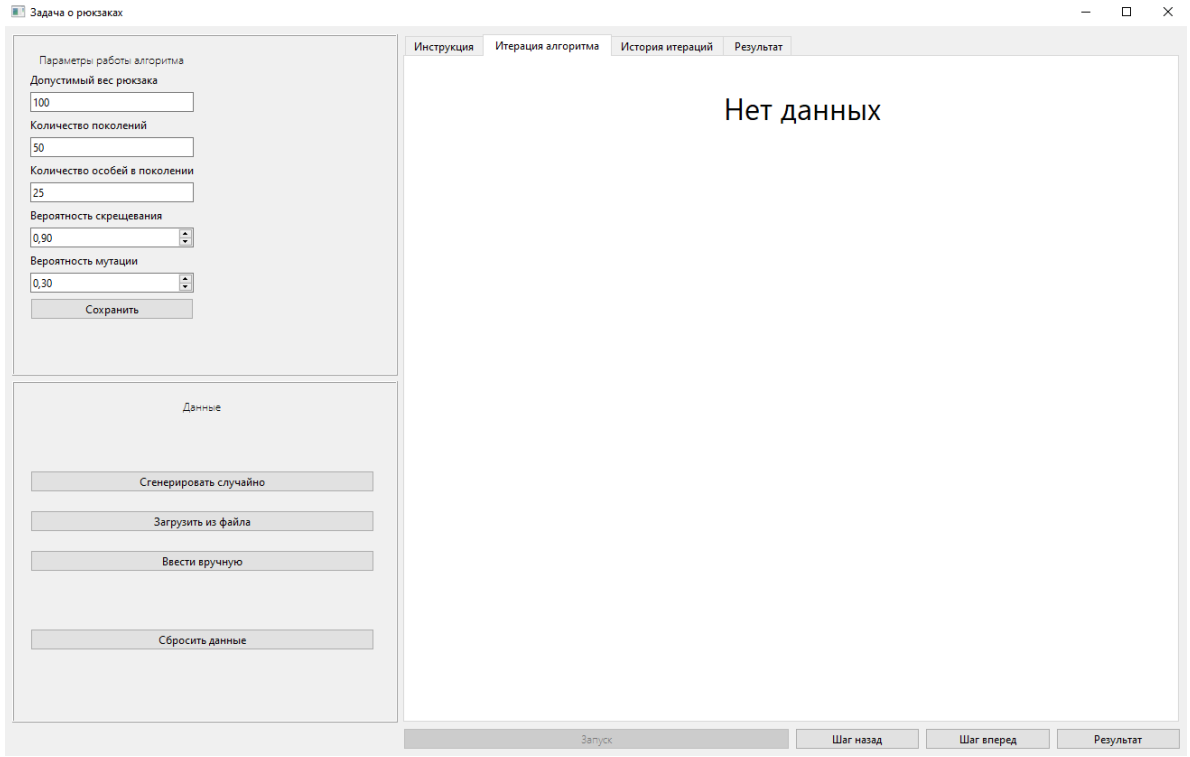

Рисунок 3 - Окно после сброса данных

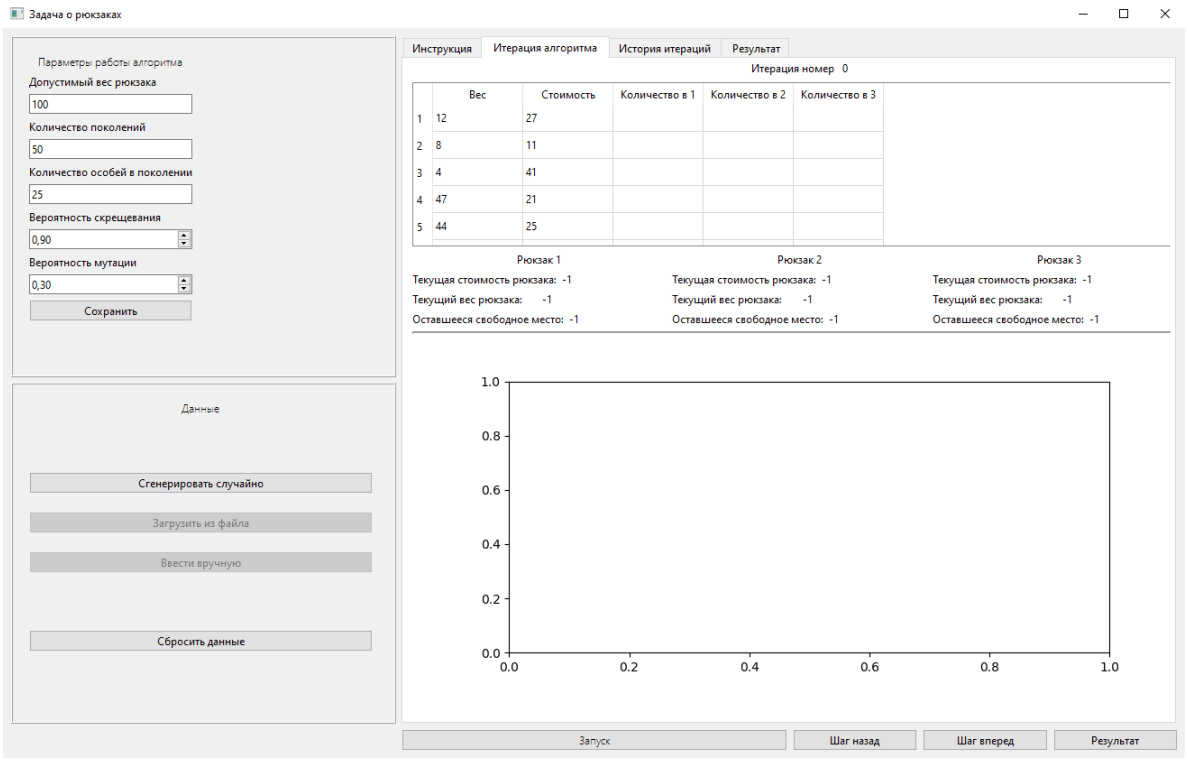

Рисунок 7 - Окно начала работы алгоритма

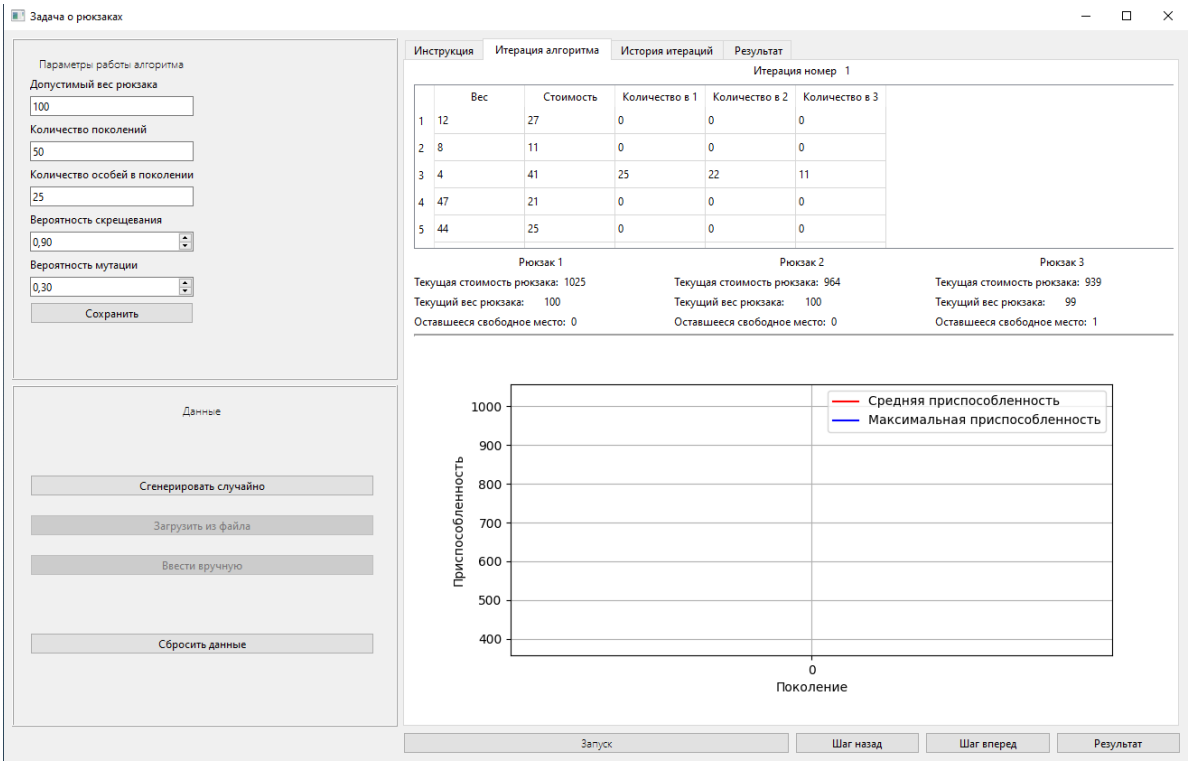

Рисунок 8 - Окно после первой итерации алгоритма,
оно же отображается после при нажатии «шаг назад» относительно состояния на рис. 9

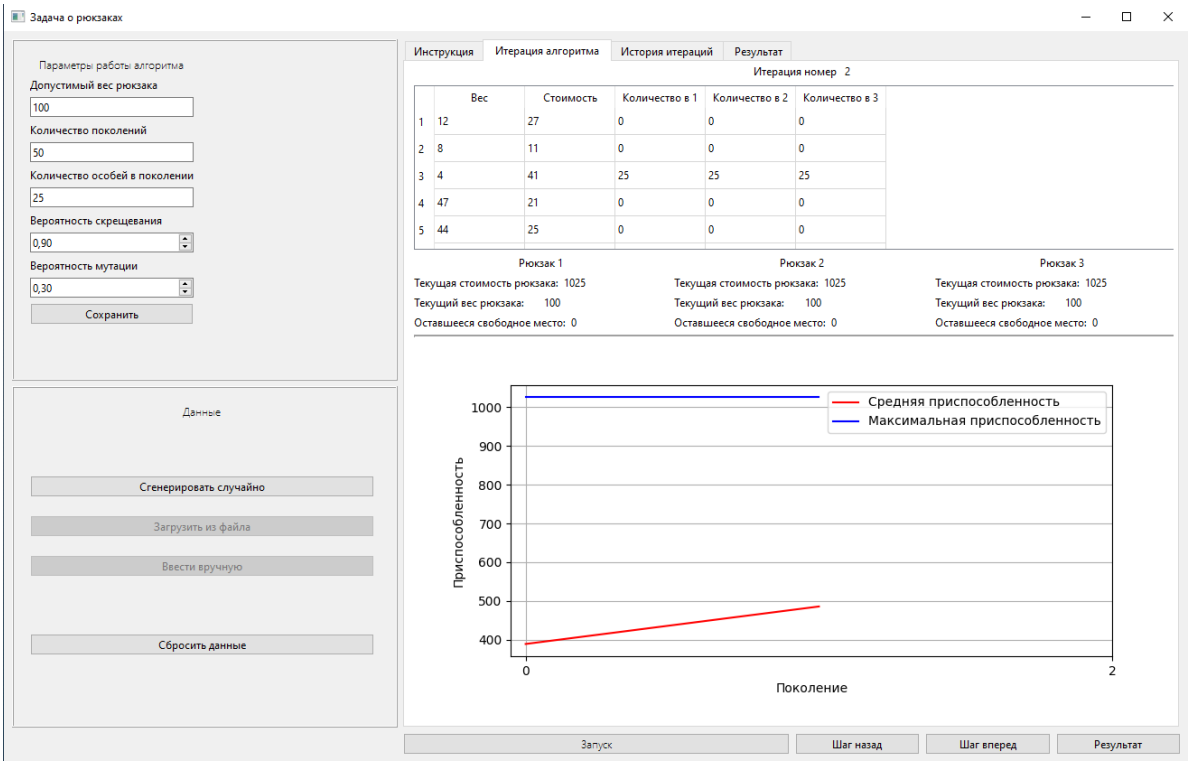

Рисунок 9 - Окно при нажатии «шаг вперед»

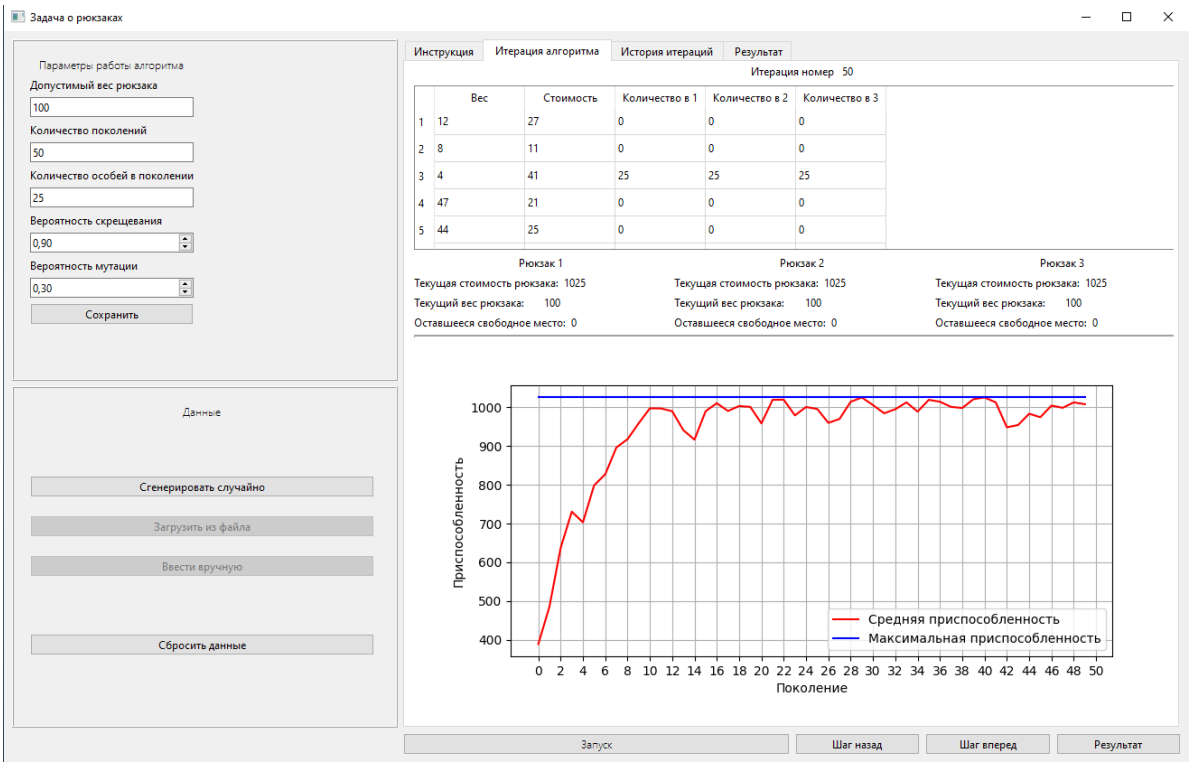

Рисунок 10 - Окно при нажатии «результат»

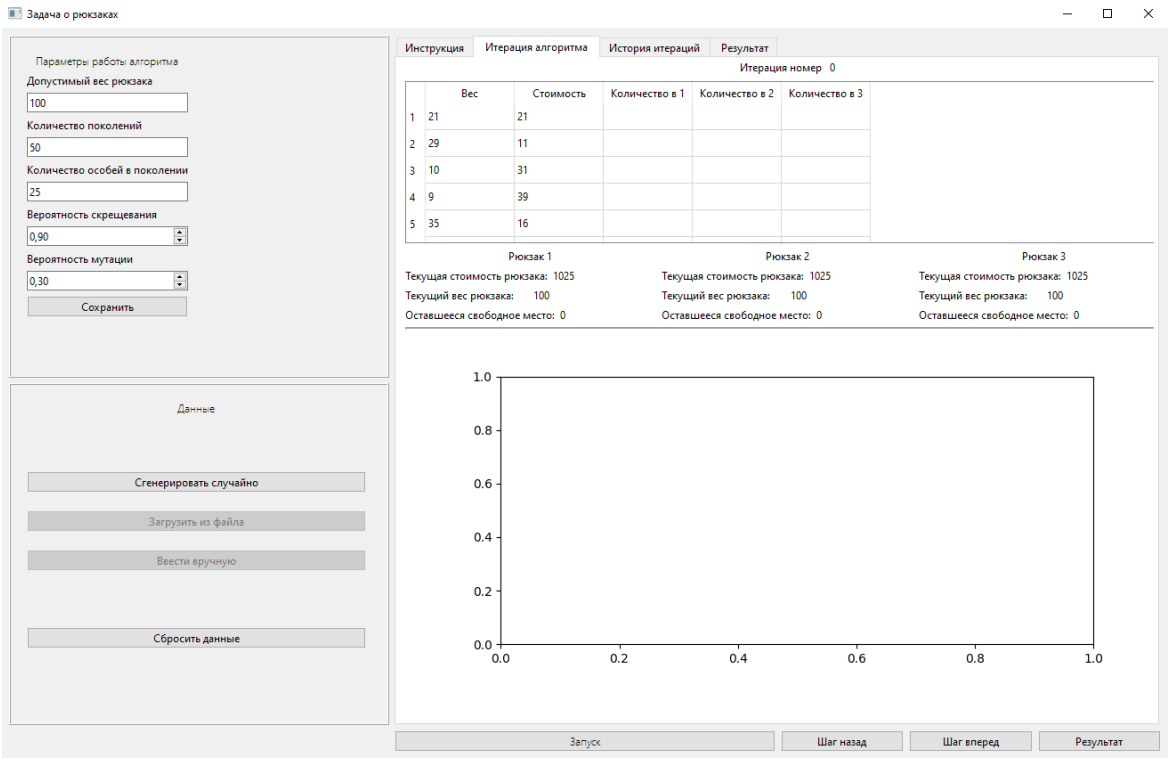

Рисунок 11 - Окно при перезапуске алгоритма

Вывод.

Изучены генетические алгоритмы. Разработан генетический алгоритм и GUI. Создана частично работающая программа, решающая задачу о неограниченном рюкзаке с помощью генетического алгоритма.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: genetic_algorithm.py

```
import numpy as np

from objects import *
import matplotlib.pyplot as plt
import random

generationNum = mutationNum = 1


class GeneticAlgorithm:
    def __init__(self, items: list[Item], parameters:
AlgorithmParameters):
        self.items = items
        self.parameters = parameters

    def generateRandomBackpack(self) -> Backpack:
        currentBackpackWeight = 0
        availableItems = self.items
        genome = [0] * len(self.items)
        while len(availableItems) != 0:
            item = random.choice(availableItems)

            if len(availableItems) == 1:
                amount = int((self.parameters.maxBackpackWeight -
currentBackpackWeight) / item.weight)
            else:
                amount = random.randint(1,
(self.parameters.maxBackpackWeight - currentBackpackWeight) //
item.weight)

            ind = self.items.index(item)
            genome[ind] += amount
            currentBackpackWeight += amount * item.weight

            availableItems = list(
                filter(lambda x: x.weight <=
self.parameters.maxBackpackWeight - currentBackpackWeight,
availableItems))
        return Backpack(genome)

    def generateRandomGeneration(self) -> Generation:
        randomGeneration = Generation([])
        for _ in range(self.parameters.amountOfIndividsPerGeneration):
            randomGeneration.append(self.generateRandomBackpack())
            randomGeneration.calculateWeight(self.items)

randomGeneration.calculateFitness(self.parameters.maxBackpackWeight,
self.items)
        return randomGeneration

    def tournamentParentsSelection(self, generation: Generation) ->
list[Backpack]:
```

```
        selectedParents = []
        while len(selectedParents) !=
self.parameters.amountOfIndividsPerGeneration:
            indexes = [i for i in range(len(generation))]
            tournamentIndexes = random.sample(indexes, 2)
            selectedParents.append(max([generation[i] for i in
tournamentIndexes]))

            global generationNum
            if len(selectedParents) < 2 and generationNum == 1:
                print("\nОтбор родителей турниром")
                individ1 = generation[tournamentIndexes[0]]
                individ2 = generation[tournamentIndexes[1]]
                print(f"Две случайно выбранные особи:")
                print(f"\t1) {individ1}")
                print(f"\t2) {individ2}")
                print(f"\tВыбираем лучшую из них: {selectedParents[-
1].genome}")

        return selectedParents

    def uniformCrossingForTwoParents(self, parents: tuple[Backpack,
Backpack]) -> list[Backpack]:
        children = [[], []]
        for j in range(len(parents[0].genome)):
            i = random.choice([0, 1])
            children[0].append(parents[i].genome[j])
            children[1].append(parents[1 - i].genome[j])

        for i in range(len(children)):
            children[i] = Backpack(children[i])
            children[i].calculateWeight(self.items)

children[i].calculateFitness(self.parameters.maxBackpackWeight,
self.items)
        return children

    def uniformParentsCrossing(self, selectedParents: list[Backpack])
-> list[Backpack]:
        producedChildren = []
        while len(producedChildren) <=
self.parameters.amountOfIndividsPerGeneration:
            parents = random.sample(selectedParents, 2)
            global generationNum
            if len(producedChildren) < 2 and generationNum == 1:
                print("\nРавномерное скрещивание особей")
                print(f"Два случайно выбранных родителя:")
                print(f"\t1) {parents[0]}")
                print(f"\t2) {parents[1]}")
            if random.random() < self.parameters.crossingProbability:
                producedChildren +=
self.uniformCrossingForTwoParents(parents)
                if len(producedChildren) == 2 and generationNum == 1:
                    print(f"Полученные дети:")
                    print(f"\t1) {producedChildren[-1]}")
                    print(f"\t1) {producedChildren[-2]}")
            elif len(producedChildren) < 2 and generationNum == 1:
```

```
                if len(producedChildren) == 2 and generationNum == 1:
                    print("Скрещивание не проводится")
        return producedChildren

    def densityMutationOneChild(self, child: Backpack) -> None:
        global mutationNum
        if mutationNum == 1:
            print("\nПлотность мутации")
            print(f"Геном до мутации:")
            print(f"{child}")

        parameter = 20
        for i in range(len(child.genome)):
            if i == mutationNum == 1:
                print(f"\tПервый ген до мутации: {child.genome[i]}")
            if random.random() < self.parameters.mutationProbability:
                delta = 0
                for j in range(parameter):
                    randVal = random.choices([1, 0], weights=[1 /
parameter, 1 - 1 / parameter])[0]
                    delta += randVal * 2 ** (-i)
                sign = random.choice([-1, 1])
                child.genome[i] = int(child.genome[i] + sign * delta *
2)
                if child.genome[i] < 0:
                    child.genome[i] = 0
                if i == mutationNum == 1:
                    print(f"\tСлучайно полученное значение, на которое
мутирует ген: {int(2 * delta)}")
                    print(f"\tЗнак мутации: {sign}")
                    print(f"\tПервый ген после мутации:
{child.genome[i]}")
            else:
                if i == mutationNum == 1:
                    print(f"\tПервый ген не мутирует")
        child.calculateWeight(self.items)
        child.calculateFitness(self.parameters.maxBackpackWeight,
self.items)
        if mutationNum == 1:
            print(f"Геном после мутации:")
            print(f"{child}")
        mutationNum = 2

    def densityChildrenMutation(self, children: list[Backpack]) ->
None:
        for i in range(len(children)):
            if random.random() < self.parameters.mutationProbability:
                self.densityMutationOneChild(children[i])

    def eliteChoice(self, selectedParents: list[Backpack],
producedChildren: list[Backpack]) -> Generation:
        allCandidates = selectedParents + producedChildren
        generation = sorted(allCandidates, key=lambda x: x.cost,
reverse=True)[
                     :int(0.1 *
self.parameters.amountOfIndividsPerGeneration)]
        global generationNum
```

```
        if generationNum == 1:
            print(f"\nЭлитарный отбор")
            print(f"Лучшие 10% родительских и детских особей:")
            self.outputBackpacks(generation)
        while len(generation) !=
self.parameters.amountOfIndividsPerGeneration:
            generation.append(random.choice(allCandidates))
        if generationNum == 1:
            print(f"Остальные 90% выбираются случайно")
            print(f"\nИтоговое новое поколение:")
            self.outputBackpacks(generation)
            print()
        return Generation(generation)

    def dynamicProgrammingSolution(self) -> Backpack:
        pass

    def drawPlot(self, maxFitness: list[int], averageFitness:
list[float]) -> None:
        x_len = self.parameters.maxAmountOfGenerations
        plt.plot(list(range(x_len)), averageFitness, 'r-')
        plt.plot(list(range(x_len)), maxFitness, 'b-')
        plt.grid()
        plt.xticks(np.arange(0, x_len + 1, 2))
        # строчка ниже все ломает, хотя она должна задавать шаг рисок
по оси oy
        # plt.yticks(np.arange(min(maxFitness), max(maxFitness)+1, 5))
        plt.xlabel('Поколение')
        plt.ylabel('Приспособленность')
        plt.show()

    def outputGenerationInfo(self, generation: Generation,
generationNumber: int):
        print(f"\nПоколение №{generationNumber}:")
        sortedGeneration = sorted(generation, key=lambda x: x.cost,
reverse=True)
        for i, solution in enumerate(sortedGeneration):
            print(f"{i + 1}) {solution.genome}")
            print(f"\tСуммарная стоимость вещей: {solution.cost}")
            print(
                f"\tСуммарный вес вещей: {solution.weight}, дельта =
{self.parameters.maxBackpackWeight - solution.weight}")
        print(f"Текущая максимальная приспособленность:
{generation.getMaxFitness()}")
        print(f"Текущая средняя приспособленность:
{generation.getAverageFitness()}")

    def outputBackpacks(self, backpacks: list[Backpack]):
        for i, backpack in enumerate(backpacks):
            print(f"{i + 1}) {backpack.genome}")
            print(f"\tСуммарная стоимость вещей: {backpack.cost}")
            print(
                f"\tСуммарный вес вещей: {backpack.weight}, дельта =
{self.parameters.maxBackpackWeight - backpack.weight}")

    def getSolution(self) -> list[IterationInfo]:
        generation = self.generateRandomGeneration()
```

```
        print(f"Начальное случайно сгенерированное поколение:")
        self.outputBackpacks(generation.backpacks)
        print()

        maxFitness = []
        averageFitness = []
        allIterations = []
        global generationNum
        for generationNumber in range(1,
self.parameters.maxAmountOfGenerations + 1):
            generationNum = generationNumber
            generation.calculateWeight(self.items)

generation.calculateFitness(self.parameters.maxBackpackWeight,
self.items)
            maxFitness.append(generation.getMaxFitness())
            averageFitness.append(generation.getAverageFitness())

allIterations.append(IterationInfo(generation.getBestBackpacks(),
maxFitness[-1], averageFitness[-1]))

            print(f"\n------------------")
            print(f"Лучшие решения поколения №{generationNumber}")
            self.outputBackpacks(generation.getBestBackpacks())

            selectedParents =
self.tournamentParentsSelection(generation)
            # if generationNumber == 1:
            #     print(f"\nОтобранные для скрещивания родители:")
            #     self.outputBackpacks(selectedParents)

            producedChildren =
self.uniformParentsCrossing(selectedParents)
            # if generationNumber == 1:
            #     print(f"\nПолученные дети:")
            #     self.outputBackpacks(producedChildren)

            self.densityChildrenMutation(producedChildren)
            # if generationNumber == 1:
            #     print(f"\nДети после мутации:")
            #     self.outputBackpacks(producedChildren)

            generation = self.eliteChoice(generation.backpacks,
producedChildren)

        self.drawPlot(maxFitness, averageFitness)
        return allIterations


# def getInput():
#     print("Введите вместимость рюкзака")
#     limitWeight = int(input())
#     items = []
#     print("Введите стоимость и вес каждой вещи с новой строки")
#     for line in sys.stdin:
#         weight, cost = line.split()
#         items.append(Item(weight, cost))
```

```
#     return items, limitWeight


if __name__ == '__main__':
    items = [Item(5, 2), Item(7, 3), Item(6, 4), Item(3, 2)]
    maxBackpackWeight = 9
    crossingProbability = 0.9
    mutationProbability = 0.2
    amountOfIndividsPerGeneration = 20
    maxAmountOfGenerations = 20

    GA = GeneticAlgorithm(items,
AlgorithmParameters(maxBackpackWeight, crossingProbability,
mutationProbability,

amountOfIndividsPerGeneration, maxAmountOfGenerations))
    GA.getSolution()
```

Название файла: objects.py

```
from typing import Iterator


class Item:
    def __init__(self, cost: int, weight: int):
        self.cost = cost
        self.weight = weight

    def __str__(self):
        return f"Вещь стоит {self.cost} и весит {self.weight}"


class Backpack:
    def __init__(self, amountOfEachItems: list[int]):
        self.genome = amountOfEachItems
        self.cost = 0
        self.weight = 0

    def __str__(self):
        return f"{self.genome}, стоимость = {self.cost}, вес =
{self.weight}"

    def __iter__(self) -> Iterator:
        return iter(self.genome)

    def __le__(self, other: 'Backpack') -> bool:
        return self.cost <= other.cost

    def __lt__(self, other: 'Backpack') -> bool:
        return self.cost < other.cost

    def __ge__(self, other: 'Backpack') -> bool:
        return self.cost >= other.cost

    def __gt__(self, other: 'Backpack') -> bool:
        return self.cost > other.cost
```

```
    def calculateWeight(self, items: list[Item]) -> None:
        self.weight = sum(items[i].weight * self.genome[i] for i in
range(len(items)))

    def calculateFitness(self, limitWeight: int, items: list[Item]) ->
None:
        sumCost = sum(items[i].cost * self.genome[i] for i in
range(len(items)))
        if self.weight <= limitWeight:
            self.cost = sumCost
        else:
            koeff = 1 - (self.weight - limitWeight) / limitWeight
            self.cost = int(sumCost * koeff)


class Generation:
    def __init__(self, backpacks: list[Backpack]):
        self.backpacks = backpacks
        self.maxFitness = 0
        self.averageFitness = 0

    def __iter__(self) -> Iterator:
        return iter(self.backpacks)

    def __len__(self) -> int:
        return len(self.backpacks)

    def __getitem__(self, key: int) -> Backpack:
        return self.backpacks[key]

    def __str__(self):
        return "\n".join(map(str, self.backpacks))

    def append(self, item: Backpack) -> None:
        self.backpacks.append(item)

    def expend(self, other: 'Generation') -> None:
        self.backpacks.extend(other)

    def remove(self, item: Backpack) -> None:
        self.backpacks.remove(item)

    def getBestBackpacks(self) -> list[Backpack]:
        sorted_backpacks = sorted(self.backpacks, key=lambda x:
x.cost, reverse=True)
        return sorted_backpacks[:3]

    def getAverageFitness(self) -> float:
        return sum(backpack.cost for backpack in self.backpacks) /
len(self.backpacks)

    def getMaxFitness(self) -> int:
        return self.getBestBackpacks()[0].cost

    def calculateWeight(self, items: list[Item]) -> None:
        for backpack in self.backpacks:
            backpack.calculateWeight(items)
```

```
    def calculateFitness(self, limitWeight: int, items: list[Item]) ->
None:
        for backpack in self.backpacks:
            backpack.calculateFitness(limitWeight, items)


class AlgorithmParameters:
    def __init__(self,
                 maxBackpackWeight: int,
                 crossingProbability: float,
                 mutationProbability: float,
                 amountOfIndividsPerGeneration: int,
                 maxAmountOfGenerations: int):
        self.maxBackpackWeight = maxBackpackWeight
        self.crossingProbability = crossingProbability
        self.mutationProbability = mutationProbability
        self.amountOfIndividsPerGeneration =
amountOfIndividsPerGeneration
        self.maxAmountOfGenerations = maxAmountOfGenerations



class IterationInfo:
    def __init__(self, bestBackpacks: list[Backpack],
currentMaxFitness: int, currentAverageFitness: float):
        self.bestBackpacks = bestBackpacks
        self.currentMaxFitness = currentMaxFitness
        self.currentAverageFitness = currentAverageFitness

# class AllInfo:
#     def __init__(self, maxBackpackWeight: int, items: list[Item]):
#         self.maxBackpackWeight = maxBackpackWeight
#         self.items = items
#         self.maxFitness = []
#         self.averageFitness = []
#
#     def appendMaxFitness(self, iteration: IterationInfo) -> None:
#         self.maxFitness.append(iteration.currentMaxFitness)
#
#     def appendAverageFitness(self, iteration: IterationInfo) ->
None:
#         self.averageFitness.append(iteration.currentAverageFitness)
#
#     def drawPlot(self) -> None:
#         pass
```

Название файла: gui.py

```
from PyQt6 import QtCore, QtGui, QtWidgets
from PyQt6.QtWidgets import QMainWindow


from UIs.UIMainWindow import Ui_MainWindow
from UIs.UIRandGenDialog import Ui_randGenDialog
```

```
    from UIs.UIHandInputUI import Ui_HandInputDialog


    class MainWindow(QMainWindow):
        def __init__(self):
            QMainWindow.__init__(self)
            self.mainWindowUI = Ui_MainWindow()
            self.mainWindowUI.setupUi(self)


    class RandGenDialog(QtWidgets.QDialog):
        def __init__(self):
            QtWidgets.QDialog.__init__(self)
            self.randGenDialogUI = Ui_randGenDialog()
            self.randGenDialogUI.setupUi(self)


    class HandInputDialog(QtWidgets.QDialog):
        def __init__(self):
            QtWidgets.QDialog.__init__(self)
            self.handInputDialogUI = Ui_HandInputDialog()
       self.handInputDialogUI.setupUi(self)
```

Название файла: UIlogic.py

```
import os
import random
import sys

from PyQt6 import QtWidgets
from PyQt6.QtGui import QIntValidator
from PyQt6.QtWidgets import QFileDialog

import gui
from src.libs.objects import *


class AllInfo:
    def __init__(self, maxBackpackWeight: int, items: list[Item]):
        self.maxBackpackWeight = maxBackpackWeight
        self.items = items
        self.maxFitness = []
        self.averageFitness = []

    def appendMaxFitness(self, iteration: CurrentIterationInfo) ->
None:
        self.maxFitness.append(iteration.currentMaxFitness)
```

```
    def appendAverageFitness(self, iteration: CurrentIterationInfo) ->
None:
        self.averageFitness.append(iteration.currentAverageFitness)

    def drawPlot(self) -> None:
        pass


items = list()


def generateRandomItems(n: int) -> list[Item]:
    for i in range(n):
        item = Item(random.randint(1, 50), random.randint(1, 50))
        items.append(item)
    return items


class Data:
    def __init__(self):
        self.algParams = AlgorithmParameters(100, 0.5, 0.5, 25, 50)

        self.randomGenerationBackpackValue = -1
        self.inputFileName = ""

        self.algNum = -1
        self.info = None


class UILogic:
    def __init__(self):
        self.mainWindow = gui.MainWindow()
        self.mainWindowUI = self.mainWindow.mainWindowUI

        self.randGenDialog = gui.RandGenDialog()
        self.randGenDialogUI = self.randGenDialog.randGenDialogUI

        self.handInputDialog = gui.HandInputDialog()
        self.handInputDialogUI =
self.handInputDialog.handInputDialogUI

        self.data = Data()

        self.mainWindowUI.startButton.setEnabled(False)
        self.mainWindowUI.iterationDataFrame.setVisible(False)
        self.connectButtons()
        self.adjustLineEdits()

    def adjustLineEdits(self):
        validator = QIntValidator(1, 999)
        self.mainWindowUI.backpackValueLE.setValidator(validator)
        self.mainWindowUI.generationAmountLE.setValidator(validator)
        self.mainWindowUI.entityAmountLE.setValidator(validator)


self.mainWindowUI.backpackValueLE.setText(str(self.data.algParams.maxB
ackpackWeight))
```

```
self.mainWindowUI.generationAmountLE.setText(str(self.data.algParams.m
axAmountOfGenerations))

self.mainWindowUI.entityAmountLE.setText(str(self.data.algParams.amoun
tOfIndividsPerGeneration))

self.mainWindowUI.crossingProbabilitySpin.setValue(self.data.algParams
.crossingProbability)

self.mainWindowUI.mutationProbabilitySpin.setValue(self.data.algParams
.mutationProbability)

    def connectButtons(self):

self.mainWindowUI.randomGenButton.clicked.connect(self.openRandomGenDi
alog)

self.mainWindowUI.browseButton.clicked.connect(self.browseEvent)

self.mainWindowUI.inputButton.clicked.connect(self.openHandInputDialog
)

self.mainWindowUI.startButton.clicked.connect(self.startButtonEvent)


self.handInputDialogUI.cancelButton.clicked.connect(self.closeHandInpu
tDialogEvent)


self.randGenDialogUI.CancelButton.clicked.connect(self.closeGenDialogE
vent)

self.randGenDialogUI.doneButton.clicked.connect(self.doneButtonEvent)


self.mainWindowUI.resetDataButton.clicked.connect(self.resetButtonEven
t)

    def openRandomGenDialog(self):
        self.randGenDialog.finished.connect(self.closeGenDialogEvent)
        self.mainWindow.setEnabled(False)

        self.randGenDialog.show()

    def switchAlgorithms(self, n: int):
        if n == 1:
            self.mainWindowUI.browseButton.setEnabled(False)
            self.mainWindowUI.inputButton.setEnabled(False)
            self.mainWindowUI.startButton.setEnabled(True)
            self.data.info =
AllInfo(self.data.algParams.maxBackpackWeight,

generateRandomItems(self.data.randomGenerationBackpackValue))
        elif n == 2:
            self.mainWindowUI.randomGenButton.setEnabled(False)
            self.mainWindowUI.inputButton.setEnabled(False)
```

```
            self.mainWindowUI.startButton.setEnabled(True)
        elif n == 3:
            self.mainWindowUI.randomGenButton.setEnabled(False)
            self.mainWindowUI.browseButton.setEnabled(False)
            self.mainWindowUI.startButton.setEnabled(True)
        elif n == -1:
            self.mainWindowUI.randomGenButton.setEnabled(True)
            self.mainWindowUI.browseButton.setEnabled(True)
            self.mainWindowUI.inputButton.setEnabled(True)
            self.mainWindowUI.startButton.setEnabled(False)

    def startButtonEvent(self):
        self.mainWindowUI.iterationTabWidget_2.setCurrentIndex(1)

        self.startAlgorithm()

    def resetButtonEvent(self):
        self.data.algNum = -1

        self.data.inputFileName = ""
        self.data.randomGenerationBackpackValue = -1

        self.switchAlgorithms(self.data.algNum)

    def openHandInputDialog(self):
        # self.switchAllButtons(False)
        self.handInputDialog.show()

self.handInputDialog.finished.connect(self.closeHandInputDialogEvent)

self.handInputDialogUI.AmountLineEdit.textEdited.connect(self.handInpu
tTextChanged)

    def handInputTextChanged(self):
        text = self.handInputDialogUI.AmountLineEdit.text()

        self.handInputDialogUI.tableWidget.setRowCount(int(text))

    def switchAllButtons(self, state: bool):
        self.mainWindowUI.inputButton.setEnabled(state)
        # self.mainWindowUI.startButton.setEnabled(state)
        self.mainWindowUI.browseButton.setEnabled(state)
        self.mainWindowUI.backButton.setEnabled(state)
        self.mainWindowUI.forwardButton.setEnabled(state)
        self.mainWindowUI.resultButton.setEnabled(state)
        self.mainWindowUI.randomGenButton.setEnabled(state)

    def closeHandInputDialogEvent(self):
        self.handInputDialog.close()
        # self.switchAllButtons(True)

    def closeGenDialogEvent(self):
        if self.randGenDialogUI.AmountLineEdit.text() != "" and
self.data.randomGenerationBackpackValue != -1:

self.randGenDialogUI.AmountLineEdit.setText(str(self.data.randomGenera
tionBackpackValue))
```

```
            self.randGenDialog.close()
            self.mainWindow.setEnabled(True)

        def doneButtonEvent(self):
            if self.randGenDialogUI.AmountLineEdit.text() != "":
                self.data.algNum = 1
                self.data.randomGenerationBackpackValue =
int(self.randGenDialogUI.AmountLineEdit.text())
                self.switchAlgorithms(self.data.algNum)

                self.randGenDialog.close()
                self.mainWindowUI.startButton.setEnabled(True)

        # Открываем диалог (выбор файла)
        def browseEvent(self):
            self.mainWindow.setEnabled(False)

            file_filter = 'Text File (*.txt)'
            file_dialog = QFileDialog
            file_name = file_dialog.getOpenFileName(
                parent=self.mainWindow,
                caption='Открыть файл',
                directory=os.getcwd(),
                filter=file_filter,
                initialFilter='Text File (*.txt)'
            )
            if file_name[0] != "":
                self.data.algNum = 2
                self.data.inputFileName = file_name[0]

            self.mainWindow.setEnabled(True)
            self.switchAlgorithms(self.data.algNum)

            print(self.data.inputFileName)

        def startAlgorithm(self):
            self.mainWindowUI.iterationDataFrame.setVisible(True)
            self.mainWindowUI.noDataLabel.setVisible(False)

self.mainWindowUI.backpackTableWidget.setRowCount(self.data.randomGene
rationBackpackValue)

self.mainWindowUI.backpackTableWidget_2.setRowCount(self.data.randomGe
nerationBackpackValue)

self.mainWindowUI.backpackTableWidget_3.setRowCount(self.data.randomGe
nerationBackpackValue)


if __name__ == "__main__":
    app = QtWidgets.QApplication([])

    logic = UILogic()
    logic.mainWindow.show()

    sys.exit(app.exec())
```

Название файла: UIHandInputUI.py

```
# Form implementation generated from reading ui file
'handInputDialogUI.ui'
#
# Created by: PyQt6 UI code generator 6.7.0
#
# WARNING: Any manual changes made to this file will be lost when
pyuic6 is
# run again.  Do not edit this file unless you know what you are
doing.


from PyQt6 import QtCore, QtGui, QtWidgets


class Ui_HandInputDialog(object):
    def setupUi(self, HandInputDialog):
        HandInputDialog.setObjectName("HandInputDialog")
        HandInputDialog.resize(280, 391)
        HandInputDialog.setSizeGripEnabled(False)
        HandInputDialog.setModal(False)
        self.gridLayout = QtWidgets.QGridLayout(HandInputDialog)
        self.gridLayout.setObjectName("gridLayout")
        self.horizontalLayout = QtWidgets.QHBoxLayout()
        self.horizontalLayout.setObjectName("horizontalLayout")
        self.cancelButton =
QtWidgets.QPushButton(parent=HandInputDialog)
        self.cancelButton.setObjectName("cancelButton")
        self.horizontalLayout.addWidget(self.cancelButton)
        self.doneButton =
QtWidgets.QPushButton(parent=HandInputDialog)
        self.doneButton.setObjectName("doneButton")
        self.horizontalLayout.addWidget(self.doneButton)
        self.gridLayout.addLayout(self.horizontalLayout, 2, 0, 1, 1)
        self.verticalLayout = QtWidgets.QVBoxLayout()
        self.verticalLayout.setObjectName("verticalLayout")
        self.amountLabel = QtWidgets.QLabel(parent=HandInputDialog)

self.amountLabel.setAlignment(QtCore.Qt.AlignmentFlag.AlignCenter)
        self.amountLabel.setObjectName("amountLabel")
        self.verticalLayout.addWidget(self.amountLabel)
        self.AmountLineEdit =
QtWidgets.QLineEdit(parent=HandInputDialog)
        self.AmountLineEdit.setObjectName("AmountLineEdit")
        self.verticalLayout.addWidget(self.AmountLineEdit)
        self.gridLayout.addLayout(self.verticalLayout, 0, 0, 1, 1)
        self.verticalLayout_2 = QtWidgets.QVBoxLayout()
        self.verticalLayout_2.setObjectName("verticalLayout_2")
        self.label_2 = QtWidgets.QLabel(parent=HandInputDialog)
        self.label_2.setObjectName("label_2")
        self.verticalLayout_2.addWidget(self.label_2)
        self.tableWidget =
QtWidgets.QTableWidget(parent=HandInputDialog)
        self.tableWidget.setGridStyle(QtCore.Qt.PenStyle.SolidLine)
        self.tableWidget.setRowCount(0)
        self.tableWidget.setColumnCount(2)
```

```
        self.tableWidget.setObjectName("tableWidget")
        item = QtWidgets.QTableWidgetItem()
        self.tableWidget.setHorizontalHeaderItem(0, item)
        item = QtWidgets.QTableWidgetItem()
        self.tableWidget.setHorizontalHeaderItem(1, item)
        self.verticalLayout_2.addWidget(self.tableWidget)
        self.gridLayout.addLayout(self.verticalLayout_2, 1, 0, 1, 1)

        self.retranslateUi(HandInputDialog)
        QtCore.QMetaObject.connectSlotsByName(HandInputDialog)

    def retranslateUi(self, HandInputDialog):
        _translate = QtCore.QCoreApplication.translate
        HandInputDialog.setWindowTitle(_translate("HandInputDialog",
"Ручной ввод"))
        self.cancelButton.setText(_translate("HandInputDialog",
"Назад"))
        self.doneButton.setText(_translate("HandInputDialog",
"Готово"))
        self.amountLabel.setText(_translate("HandInputDialog",
"Введите количество предметов"))

self.AmountLineEdit.setPlaceholderText(_translate("HandInputDialog",
"Кол-во предметов"))
        self.label_2.setText(_translate("HandInputDialog", "Введите
соответсвующую информацию"))
        self.tableWidget.setSortingEnabled(True)
        item = self.tableWidget.horizontalHeaderItem(0)
        item.setText(_translate("HandInputDialog", "Вес"))
        item = self.tableWidget.horizontalHeaderItem(1)
        item.setText(_translate("HandInputDialog", "Цена"))


if __name__ == "__main__":
    import sys
    app = QtWidgets.QApplication(sys.argv)
    HandInputDialog = QtWidgets.QDialog()
    ui = Ui_HandInputDialog()
    ui.setupUi(HandInputDialog)
    HandInputDialog.show()
    sys.exit(app.exec())
```

Название файла: UIMainWindow.py

```
# Form implementation generated from reading ui file
'backpackProblemUI.ui'
#
# Created by: PyQt6 UI code generator 6.7.0
#
# WARNING: Any manual changes made to this file will be lost when
pyuic6 is
# run again.  Do not edit this file unless you know what you are
doing.


from PyQt6 import QtCore, QtGui, QtWidgets
```

```
class Ui_MainWindow(object):
    def setupUi(self, MainWindow):
        MainWindow.setObjectName("MainWindow")
        MainWindow.resize(1312, 810)
        self.wind = QtWidgets.QWidget(parent=MainWindow)
        self.wind.setEnabled(True)
        self.wind.setObjectName("wind")
        self.gridLayout = QtWidgets.QGridLayout(self.wind)
        self.gridLayout.setObjectName("gridLayout")
        self.horizontalLayout = QtWidgets.QHBoxLayout()
        self.horizontalLayout.setObjectName("horizontalLayout")
        self.backButton = QtWidgets.QPushButton(parent=self.wind)
        self.backButton.setObjectName("backButton")
        self.horizontalLayout.addWidget(self.backButton)
        self.forwardButton = QtWidgets.QPushButton(parent=self.wind)
        self.forwardButton.setObjectName("forwardButton")
        self.horizontalLayout.addWidget(self.forwardButton)
        self.resultButton = QtWidgets.QPushButton(parent=self.wind)
        self.resultButton.setObjectName("resultButton")
        self.horizontalLayout.addWidget(self.resultButton)
        self.gridLayout.addLayout(self.horizontalLayout, 2, 2, 1, 1)
        self.verticalLayout = QtWidgets.QVBoxLayout()
        self.verticalLayout.setObjectName("verticalLayout")
        self.paramsDataFrame = QtWidgets.QFrame(parent=self.wind)
        self.paramsDataFrame.setFrameShape(QtWidgets.QFrame.Shape.Box)

self.paramsDataFrame.setFrameShadow(QtWidgets.QFrame.Shadow.Raised)
        self.paramsDataFrame.setObjectName("paramsDataFrame")
        self.verticalLayoutWidget =
QtWidgets.QWidget(parent=self.paramsDataFrame)
        self.verticalLayoutWidget.setGeometry(QtCore.QRect(0, 0, 421,
361))

self.verticalLayoutWidget.setObjectName("verticalLayoutWidget")
        self.algParams =
QtWidgets.QVBoxLayout(self.verticalLayoutWidget)

self.algParams.setSizeConstraint(QtWidgets.QLayout.SizeConstraint.SetF
ixedSize)
        self.algParams.setContentsMargins(20, 20, 20, 20)
        self.algParams.setObjectName("algParams")
        self.ParamsLabel =
QtWidgets.QLabel(parent=self.verticalLayoutWidget)
        self.ParamsLabel.setEnabled(True)
        font = QtGui.QFont()
        font.setBold(True)
        font.setUnderline(False)
        font.setWeight(75)
        self.ParamsLabel.setFont(font)

self.ParamsLabel.setLayoutDirection(QtCore.Qt.LayoutDirection.LeftToRi
ght)
        self.ParamsLabel.setTextFormat(QtCore.Qt.TextFormat.AutoText)
        self.ParamsLabel.setScaledContents(False)

self.ParamsLabel.setAlignment(QtCore.Qt.AlignmentFlag.AlignCenter)
```

```
        self.ParamsLabel.setObjectName("ParamsLabel")
        self.algParams.addWidget(self.ParamsLabel)
        self.backpackValueLabel =
QtWidgets.QLabel(parent=self.verticalLayoutWidget)
        self.backpackValueLabel.setObjectName("backpackValueLabel")
        self.algParams.addWidget(self.backpackValueLabel)
        self.backpackValueLE =
QtWidgets.QLineEdit(parent=self.verticalLayoutWidget)
        self.backpackValueLE.setInputMask("")
        self.backpackValueLE.setText("")
        self.backpackValueLE.setClearButtonEnabled(False)
        self.backpackValueLE.setObjectName("backpackValueLE")
        self.algParams.addWidget(self.backpackValueLE)
        self.genAmountLabel =
QtWidgets.QLabel(parent=self.verticalLayoutWidget)
        self.genAmountLabel.setObjectName("genAmountLabel")
        self.algParams.addWidget(self.genAmountLabel)
        self.generationAmountLE =
QtWidgets.QLineEdit(parent=self.verticalLayoutWidget)
        self.generationAmountLE.setObjectName("generationAmountLE")
        self.algParams.addWidget(self.generationAmountLE)
        self.entityAmountLabel =
QtWidgets.QLabel(parent=self.verticalLayoutWidget)
        self.entityAmountLabel.setObjectName("entityAmountLabel")
        self.algParams.addWidget(self.entityAmountLabel)
        self.entityAmountLE =
QtWidgets.QLineEdit(parent=self.verticalLayoutWidget)
        self.entityAmountLE.setObjectName("entityAmountLE")
        self.algParams.addWidget(self.entityAmountLE)
        self.probabilityCrossingLabel =
QtWidgets.QLabel(parent=self.verticalLayoutWidget)

self.probabilityCrossingLabel.setObjectName("probabilityCrossingLabel"
)
        self.algParams.addWidget(self.probabilityCrossingLabel)
        self.crossingProbabilitySpin =
QtWidgets.QDoubleSpinBox(parent=self.verticalLayoutWidget)
        self.crossingProbabilitySpin.setMaximum(1.0)

self.crossingProbabilitySpin.setObjectName("crossingProbabilitySpin")
        self.algParams.addWidget(self.crossingProbabilitySpin)
        self.probabilityCrossingLabel_2 =
QtWidgets.QLabel(parent=self.verticalLayoutWidget)

self.probabilityCrossingLabel_2.setObjectName("probabilityCrossingLabe
l_2")
        self.algParams.addWidget(self.probabilityCrossingLabel_2)
        self.mutationProbabilitySpin =
QtWidgets.QDoubleSpinBox(parent=self.verticalLayoutWidget)
        self.mutationProbabilitySpin.setMaximum(1.0)

self.mutationProbabilitySpin.setObjectName("mutationProbabilitySpin")
        self.algParams.addWidget(self.mutationProbabilitySpin)
        self.verticalLayout.addWidget(self.paramsDataFrame)
        self.dataFrame = QtWidgets.QFrame(parent=self.wind)
        self.dataFrame.setFrameShape(QtWidgets.QFrame.Shape.Box)
        self.dataFrame.setFrameShadow(QtWidgets.QFrame.Shadow.Raised)
```

```
        self.dataFrame.setMidLineWidth(0)
        self.dataFrame.setObjectName("dataFrame")
        self.verticalLayoutWidget_2 =
QtWidgets.QWidget(parent=self.dataFrame)
        self.verticalLayoutWidget_2.setGeometry(QtCore.QRect(0, 0,
421, 361))

self.verticalLayoutWidget_2.setObjectName("verticalLayoutWidget_2")
        self.DataLayout =
QtWidgets.QVBoxLayout(self.verticalLayoutWidget_2)
        self.DataLayout.setContentsMargins(20, 20, 20, 20)
        self.DataLayout.setSpacing(20)
        self.DataLayout.setObjectName("DataLayout")
        self.DataLabel =
QtWidgets.QLabel(parent=self.verticalLayoutWidget_2)
        self.DataLabel.setEnabled(True)
        font = QtGui.QFont()
        font.setBold(True)
        font.setWeight(75)
        self.DataLabel.setFont(font)

self.DataLabel.setLayoutDirection(QtCore.Qt.LayoutDirection.LeftToRigh
t)
        self.DataLabel.setTextFormat(QtCore.Qt.TextFormat.AutoText)
        self.DataLabel.setScaledContents(False)

self.DataLabel.setAlignment(QtCore.Qt.AlignmentFlag.AlignCenter)
        self.DataLabel.setObjectName("DataLabel")
        self.DataLayout.addWidget(self.DataLabel)
        spacerItem = QtWidgets.QSpacerItem(20, 40,
QtWidgets.QSizePolicy.Policy.Minimum,
QtWidgets.QSizePolicy.Policy.Expanding)
        self.DataLayout.addItem(spacerItem)
        self.randomGenButton =
QtWidgets.QPushButton(parent=self.verticalLayoutWidget_2)
        self.randomGenButton.setObjectName("randomGenButton")
        self.DataLayout.addWidget(self.randomGenButton)
        self.browseButton =
QtWidgets.QPushButton(parent=self.verticalLayoutWidget_2)
        self.browseButton.setObjectName("browseButton")
        self.DataLayout.addWidget(self.browseButton)
        self.inputButton =
QtWidgets.QPushButton(parent=self.verticalLayoutWidget_2)
        self.inputButton.setObjectName("inputButton")
        self.DataLayout.addWidget(self.inputButton)
        spacerItem1 = QtWidgets.QSpacerItem(20, 40,
QtWidgets.QSizePolicy.Policy.Minimum,
QtWidgets.QSizePolicy.Policy.Expanding)
        self.DataLayout.addItem(spacerItem1)
        self.resetDataButton =
QtWidgets.QPushButton(parent=self.verticalLayoutWidget_2)
        self.resetDataButton.setObjectName("resetDataButton")
        self.DataLayout.addWidget(self.resetDataButton)
        spacerItem2 = QtWidgets.QSpacerItem(20, 40,
QtWidgets.QSizePolicy.Policy.Minimum,
QtWidgets.QSizePolicy.Policy.Expanding)
        self.DataLayout.addItem(spacerItem2)
```

```
        self.verticalLayout.addWidget(self.dataFrame)
        self.gridLayout.addLayout(self.verticalLayout, 0, 0, 1, 1)
        self.startButton = QtWidgets.QPushButton(parent=self.wind)
        self.startButton.setEnabled(False)
        font = QtGui.QFont()
        font.setBold(True)
        font.setUnderline(False)
        font.setWeight(75)
        self.startButton.setFont(font)
        self.startButton.setObjectName("startButton")
        self.gridLayout.addWidget(self.startButton, 2, 1, 1, 1)
        self.iterationTabWidget_2 =
QtWidgets.QTabWidget(parent=self.wind)

self.iterationTabWidget_2.setTabPosition(QtWidgets.QTabWidget.TabPosit
ion.North)
        self.iterationTabWidget_2.setUsesScrollButtons(False)
        self.iterationTabWidget_2.setMovable(True)

self.iterationTabWidget_2.setObjectName("iterationTabWidget_2")
        self.instructionTab_2 = QtWidgets.QWidget()
        self.instructionTab_2.setObjectName("instructionTab_2")
        self.iterationTabWidget_2.addTab(self.instructionTab_2, "")
        self.iterationTab_2 = QtWidgets.QWidget()
        self.iterationTab_2.setObjectName("iterationTab_2")
        self.noDataLabel =
QtWidgets.QLabel(parent=self.iterationTab_2)
        self.noDataLabel.setGeometry(QtCore.QRect(290, 20, 301, 71))
        font = QtGui.QFont()
        font.setPointSize(25)
        self.noDataLabel.setFont(font)

self.noDataLabel.setAlignment(QtCore.Qt.AlignmentFlag.AlignCenter)
        self.noDataLabel.setObjectName("noDataLabel")
        self.iterationDataFrame =
QtWidgets.QFrame(parent=self.iterationTab_2)
        self.iterationDataFrame.setGeometry(QtCore.QRect(0, 0, 851,
721))

self.iterationDataFrame.setFrameShape(QtWidgets.QFrame.Shape.StyledPan
el)

self.iterationDataFrame.setFrameShadow(QtWidgets.QFrame.Shadow.Raised)
        self.iterationDataFrame.setObjectName("iterationDataFrame")
        self.verticalLayoutWidget_3 =
QtWidgets.QWidget(parent=self.iterationDataFrame)
        self.verticalLayoutWidget_3.setGeometry(QtCore.QRect(10, 0,
841, 701))

self.verticalLayoutWidget_3.setObjectName("verticalLayoutWidget_3")
        self.iterationTabLayout =
QtWidgets.QVBoxLayout(self.verticalLayoutWidget_3)
        self.iterationTabLayout.setContentsMargins(0, 0, 0, 0)
        self.iterationTabLayout.setObjectName("iterationTabLayout")
        self.iterationLabelLayout = QtWidgets.QHBoxLayout()

self.iterationLabelLayout.setObjectName("iterationLabelLayout")
```

```
        spacerItem3 = QtWidgets.QSpacerItem(40, 20,
QtWidgets.QSizePolicy.Policy.Expanding,
QtWidgets.QSizePolicy.Policy.Minimum)
        self.iterationLabelLayout.addItem(spacerItem3)
        self.iterationLabel =
QtWidgets.QLabel(parent=self.verticalLayoutWidget_3)
        self.iterationLabel.setObjectName("iterationLabel")
        self.iterationLabelLayout.addWidget(self.iterationLabel)
        self.iterationNumLabel =
QtWidgets.QLabel(parent=self.verticalLayoutWidget_3)
        self.iterationNumLabel.setObjectName("iterationNumLabel")
        self.iterationLabelLayout.addWidget(self.iterationNumLabel)
        spacerItem4 = QtWidgets.QSpacerItem(40, 20,
QtWidgets.QSizePolicy.Policy.Expanding,
QtWidgets.QSizePolicy.Policy.Minimum)
        self.iterationLabelLayout.addItem(spacerItem4)
        self.iterationTabLayout.addLayout(self.iterationLabelLayout)
        spacerItem5 = QtWidgets.QSpacerItem(20, 40,
QtWidgets.QSizePolicy.Policy.Minimum,
QtWidgets.QSizePolicy.Policy.Expanding)
        self.iterationTabLayout.addItem(spacerItem5)
        self.horizontalLayout_8 = QtWidgets.QHBoxLayout()
        self.horizontalLayout_8.setObjectName("horizontalLayout_8")
        self.backpackTableWidget =
QtWidgets.QTableWidget(parent=self.verticalLayoutWidget_3)
        sizePolicy =
QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Policy.Maximum,
QtWidgets.QSizePolicy.Policy.Maximum)
        sizePolicy.setHorizontalStretch(0)
        sizePolicy.setVerticalStretch(0)

sizePolicy.setHeightForWidth(self.backpackTableWidget.sizePolicy().has
HeightForWidth())
        self.backpackTableWidget.setSizePolicy(sizePolicy)
        self.backpackTableWidget.setMaximumSize(QtCore.QSize(450,
450))
        self.backpackTableWidget.setRowCount(20)
        self.backpackTableWidget.setColumnCount(3)
        self.backpackTableWidget.setObjectName("backpackTableWidget")
        item = QtWidgets.QTableWidgetItem()
        self.backpackTableWidget.setHorizontalHeaderItem(0, item)
        item = QtWidgets.QTableWidgetItem()
        self.backpackTableWidget.setHorizontalHeaderItem(1, item)
        item = QtWidgets.QTableWidgetItem()
        self.backpackTableWidget.setHorizontalHeaderItem(2, item)
        self.horizontalLayout_8.addWidget(self.backpackTableWidget)
        self.verticalLayout_3 = QtWidgets.QVBoxLayout()
        self.verticalLayout_3.setObjectName("verticalLayout_3")
        self.horizontalLayout_2 = QtWidgets.QHBoxLayout()
        self.horizontalLayout_2.setObjectName("horizontalLayout_2")
        self.textCurWeightLabel =
QtWidgets.QLabel(parent=self.verticalLayoutWidget_3)
        self.textCurWeightLabel.setObjectName("textCurWeightLabel")
        self.horizontalLayout_2.addWidget(self.textCurWeightLabel)
        self.curWeightLabel =
QtWidgets.QLabel(parent=self.verticalLayoutWidget_3)
        self.curWeightLabel.setObjectName("curWeightLabel")
```

```
        self.horizontalLayout_2.addWidget(self.curWeightLabel)
        self.verticalLayout_3.addLayout(self.horizontalLayout_2)
        self.backpackData_1 = QtWidgets.QHBoxLayout()
        self.backpackData_1.setObjectName("backpackData_1")
        self.textFreeSpacLabel =
QtWidgets.QLabel(parent=self.verticalLayoutWidget_3)
        self.textFreeSpacLabel.setObjectName("textFreeSpacLabel")
        self.backpackData_1.addWidget(self.textFreeSpacLabel)
        self.freeSpaveLabel =
QtWidgets.QLabel(parent=self.verticalLayoutWidget_3)
        self.freeSpaveLabel.setObjectName("freeSpaveLabel")
        self.backpackData_1.addWidget(self.freeSpaveLabel)
        self.verticalLayout_3.addLayout(self.backpackData_1)
        self.horizontalLayout_11 = QtWidgets.QHBoxLayout()
        self.horizontalLayout_11.setObjectName("horizontalLayout_11")
        self.textCurBPCostLabel =
QtWidgets.QLabel(parent=self.verticalLayoutWidget_3)
        self.textCurBPCostLabel.setObjectName("textCurBPCostLabel")
        self.horizontalLayout_11.addWidget(self.textCurBPCostLabel)
        self.curBPCostLabel =
QtWidgets.QLabel(parent=self.verticalLayoutWidget_3)
        self.curBPCostLabel.setObjectName("curBPCostLabel")
        self.horizontalLayout_11.addWidget(self.curBPCostLabel)
        self.verticalLayout_3.addLayout(self.horizontalLayout_11)
        self.horizontalLayout_8.addLayout(self.verticalLayout_3)
        self.iterationTabLayout.addLayout(self.horizontalLayout_8)
        self.horizontalLayout_14 = QtWidgets.QHBoxLayout()
        self.horizontalLayout_14.setObjectName("horizontalLayout_14")
        self.backpackTableWidget_2 =
QtWidgets.QTableWidget(parent=self.verticalLayoutWidget_3)
        sizePolicy =
QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Policy.Maximum,
QtWidgets.QSizePolicy.Policy.Maximum)
        sizePolicy.setHorizontalStretch(0)
        sizePolicy.setVerticalStretch(0)

sizePolicy.setHeightForWidth(self.backpackTableWidget_2.sizePolicy().h
asHeightForWidth())
        self.backpackTableWidget_2.setSizePolicy(sizePolicy)
        self.backpackTableWidget_2.setMaximumSize(QtCore.QSize(450,
450))
        self.backpackTableWidget_2.setRowCount(20)
        self.backpackTableWidget_2.setColumnCount(3)

self.backpackTableWidget_2.setObjectName("backpackTableWidget_2")
        item = QtWidgets.QTableWidgetItem()
        self.backpackTableWidget_2.setHorizontalHeaderItem(0, item)
        item = QtWidgets.QTableWidgetItem()
        self.backpackTableWidget_2.setHorizontalHeaderItem(1, item)
        item = QtWidgets.QTableWidgetItem()
        self.backpackTableWidget_2.setHorizontalHeaderItem(2, item)
        self.horizontalLayout_14.addWidget(self.backpackTableWidget_2)
        self.verticalLayout_5 = QtWidgets.QVBoxLayout()
        self.verticalLayout_5.setObjectName("verticalLayout_5")
        self.horizontalLayout_15 = QtWidgets.QHBoxLayout()
        self.horizontalLayout_15.setObjectName("horizontalLayout_15")
        self.textCurWeightLabel_2 =
```

```
QtWidgets.QLabel(parent=self.verticalLayoutWidget_3)

self.textCurWeightLabel_2.setObjectName("textCurWeightLabel_2")
        self.horizontalLayout_15.addWidget(self.textCurWeightLabel_2)
        self.curWeightLabel_2 =
QtWidgets.QLabel(parent=self.verticalLayoutWidget_3)
        self.curWeightLabel_2.setObjectName("curWeightLabel_2")
        self.horizontalLayout_15.addWidget(self.curWeightLabel_2)
        self.verticalLayout_5.addLayout(self.horizontalLayout_15)
        self.backpackData_4 = QtWidgets.QHBoxLayout()
        self.backpackData_4.setObjectName("backpackData_4")
        self.textFreeSpacLabel_2 =
QtWidgets.QLabel(parent=self.verticalLayoutWidget_3)
        self.textFreeSpacLabel_2.setObjectName("textFreeSpacLabel_2")
        self.backpackData_4.addWidget(self.textFreeSpacLabel_2)
        self.freeSpaveLabel_2 =
QtWidgets.QLabel(parent=self.verticalLayoutWidget_3)
        self.freeSpaveLabel_2.setObjectName("freeSpaveLabel_2")
        self.backpackData_4.addWidget(self.freeSpaveLabel_2)
        self.verticalLayout_5.addLayout(self.backpackData_4)
        self.horizontalLayout_16 = QtWidgets.QHBoxLayout()
        self.horizontalLayout_16.setObjectName("horizontalLayout_16")
        self.textCurBPCostLabel_2 =
QtWidgets.QLabel(parent=self.verticalLayoutWidget_3)

self.textCurBPCostLabel_2.setObjectName("textCurBPCostLabel_2")
        self.horizontalLayout_16.addWidget(self.textCurBPCostLabel_2)
        self.curBPCostLabel_2 =
QtWidgets.QLabel(parent=self.verticalLayoutWidget_3)
        self.curBPCostLabel_2.setObjectName("curBPCostLabel_2")
        self.horizontalLayout_16.addWidget(self.curBPCostLabel_2)
        self.verticalLayout_5.addLayout(self.horizontalLayout_16)
        self.horizontalLayout_14.addLayout(self.verticalLayout_5)
        self.iterationTabLayout.addLayout(self.horizontalLayout_14)
        self.horizontalLayout_17 = QtWidgets.QHBoxLayout()
        self.horizontalLayout_17.setObjectName("horizontalLayout_17")
        self.backpackTableWidget_3 =
QtWidgets.QTableWidget(parent=self.verticalLayoutWidget_3)
        sizePolicy =
QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Policy.Maximum,
QtWidgets.QSizePolicy.Policy.Maximum)
        sizePolicy.setHorizontalStretch(0)
        sizePolicy.setVerticalStretch(0)

sizePolicy.setHeightForWidth(self.backpackTableWidget_3.sizePolicy().h
asHeightForWidth())
        self.backpackTableWidget_3.setSizePolicy(sizePolicy)
        self.backpackTableWidget_3.setMaximumSize(QtCore.QSize(450,
450))
        self.backpackTableWidget_3.setRowCount(20)
        self.backpackTableWidget_3.setColumnCount(3)

self.backpackTableWidget_3.setObjectName("backpackTableWidget_3")
        item = QtWidgets.QTableWidgetItem()
        self.backpackTableWidget_3.setHorizontalHeaderItem(0, item)
        item = QtWidgets.QTableWidgetItem()
        self.backpackTableWidget_3.setHorizontalHeaderItem(1, item)
```

```
        item = QtWidgets.QTableWidgetItem()
        self.backpackTableWidget_3.setHorizontalHeaderItem(2, item)
        self.horizontalLayout_17.addWidget(self.backpackTableWidget_3)
        self.verticalLayout_6 = QtWidgets.QVBoxLayout()
        self.verticalLayout_6.setObjectName("verticalLayout_6")
        self.horizontalLayout_18 = QtWidgets.QHBoxLayout()
        self.horizontalLayout_18.setObjectName("horizontalLayout_18")
        self.textCurWeightLabel_3 =
QtWidgets.QLabel(parent=self.verticalLayoutWidget_3)

self.textCurWeightLabel_3.setObjectName("textCurWeightLabel_3")
        self.horizontalLayout_18.addWidget(self.textCurWeightLabel_3)
        self.curWeightLabel_3 =
QtWidgets.QLabel(parent=self.verticalLayoutWidget_3)
        self.curWeightLabel_3.setObjectName("curWeightLabel_3")
        self.horizontalLayout_18.addWidget(self.curWeightLabel_3)
        self.verticalLayout_6.addLayout(self.horizontalLayout_18)
        self.backpackData_5 = QtWidgets.QHBoxLayout()
        self.backpackData_5.setObjectName("backpackData_5")
        self.textFreeSpacLabel_3 =
QtWidgets.QLabel(parent=self.verticalLayoutWidget_3)
        self.textFreeSpacLabel_3.setObjectName("textFreeSpacLabel_3")
        self.backpackData_5.addWidget(self.textFreeSpacLabel_3)
        self.freeSpaveLabel_3 =
QtWidgets.QLabel(parent=self.verticalLayoutWidget_3)
        self.freeSpaveLabel_3.setObjectName("freeSpaveLabel_3")
        self.backpackData_5.addWidget(self.freeSpaveLabel_3)
        self.verticalLayout_6.addLayout(self.backpackData_5)
        self.horizontalLayout_19 = QtWidgets.QHBoxLayout()
        self.horizontalLayout_19.setObjectName("horizontalLayout_19")
        self.textCurBPCostLabel_3 =
QtWidgets.QLabel(parent=self.verticalLayoutWidget_3)

self.textCurBPCostLabel_3.setObjectName("textCurBPCostLabel_3")
        self.horizontalLayout_19.addWidget(self.textCurBPCostLabel_3)
        self.curBPCostLabel_3 =
QtWidgets.QLabel(parent=self.verticalLayoutWidget_3)
        self.curBPCostLabel_3.setObjectName("curBPCostLabel_3")
        self.horizontalLayout_19.addWidget(self.curBPCostLabel_3)
        self.verticalLayout_6.addLayout(self.horizontalLayout_19)
        self.horizontalLayout_17.addLayout(self.verticalLayout_6)
        self.iterationTabLayout.addLayout(self.horizontalLayout_17)
        self.line =
QtWidgets.QFrame(parent=self.verticalLayoutWidget_3)
        self.line.setLineWidth(2)
        self.line.setMidLineWidth(1)
        self.line.setFrameShape(QtWidgets.QFrame.Shape.HLine)
        self.line.setFrameShadow(QtWidgets.QFrame.Shadow.Sunken)
        self.line.setObjectName("line")
        self.iterationTabLayout.addWidget(self.line)
        self.iterationDataLabelLayout = QtWidgets.QHBoxLayout()
        self.iterationDataLabelLayout.setContentsMargins(10, -1, 10, -
1)

self.iterationDataLabelLayout.setObjectName("iterationDataLabelLayout"
)
        self.textSumWeightLabel =
```

```
QtWidgets.QLabel(parent=self.verticalLayoutWidget_3)
        self.textSumWeightLabel.setObjectName("textSumWeightLabel")

self.iterationDataLabelLayout.addWidget(self.textSumWeightLabel)
        self.sumWeightLabel =
QtWidgets.QLabel(parent=self.verticalLayoutWidget_3)
        self.sumWeightLabel.setObjectName("sumWeightLabel")
        self.iterationDataLabelLayout.addWidget(self.sumWeightLabel)
        spacerItem6 = QtWidgets.QSpacerItem(40, 20,
QtWidgets.QSizePolicy.Policy.Expanding,
QtWidgets.QSizePolicy.Policy.Minimum)
        self.iterationDataLabelLayout.addItem(spacerItem6)
        self.textDeltaWeightLabel =
QtWidgets.QLabel(parent=self.verticalLayoutWidget_3)

self.textDeltaWeightLabel.setObjectName("textDeltaWeightLabel")

self.iterationDataLabelLayout.addWidget(self.textDeltaWeightLabel)
        self.deltaWeightLabel =
QtWidgets.QLabel(parent=self.verticalLayoutWidget_3)
        self.deltaWeightLabel.setObjectName("deltaWeightLabel")
        self.iterationDataLabelLayout.addWidget(self.deltaWeightLabel)
        spacerItem7 = QtWidgets.QSpacerItem(40, 20,
QtWidgets.QSizePolicy.Policy.Expanding,
QtWidgets.QSizePolicy.Policy.Minimum)
        self.iterationDataLabelLayout.addItem(spacerItem7)
        self.textSumCostLabel =
QtWidgets.QLabel(parent=self.verticalLayoutWidget_3)
        self.textSumCostLabel.setObjectName("textSumCostLabel")
        self.iterationDataLabelLayout.addWidget(self.textSumCostLabel)
        self.sumCostLabel =
QtWidgets.QLabel(parent=self.verticalLayoutWidget_3)
        self.sumCostLabel.setObjectName("sumCostLabel")
        self.iterationDataLabelLayout.addWidget(self.sumCostLabel)
        spacerItem8 = QtWidgets.QSpacerItem(40, 20,
QtWidgets.QSizePolicy.Policy.Expanding,
QtWidgets.QSizePolicy.Policy.Minimum)
        self.iterationDataLabelLayout.addItem(spacerItem8)
        self.textDeltaMaxCostLabel =
QtWidgets.QLabel(parent=self.verticalLayoutWidget_3)

self.textDeltaMaxCostLabel.setObjectName("textDeltaMaxCostLabel")

self.iterationDataLabelLayout.addWidget(self.textDeltaMaxCostLabel)
        self.deltaMaxCostLabel =
QtWidgets.QLabel(parent=self.verticalLayoutWidget_3)
        self.deltaMaxCostLabel.setObjectName("deltaMaxCostLabel")

self.iterationDataLabelLayout.addWidget(self.deltaMaxCostLabel)

self.iterationTabLayout.addLayout(self.iterationDataLabelLayout)
        self.graphicsView =
QtWidgets.QGraphicsView(parent=self.verticalLayoutWidget_3)
        self.graphicsView.setObjectName("graphicsView")
        self.iterationTabLayout.addWidget(self.graphicsView)
        self.iterationTabWidget_2.addTab(self.iterationTab_2, "")
        self.historyOfIterationsTab_2 = QtWidgets.QWidget()
```

```
self.historyOfIterationsTab_2.setObjectName("historyOfIterationsTab_2"
)

self.iterationTabWidget_2.addTab(self.historyOfIterationsTab_2, "")
        self.resultTab_2 = QtWidgets.QWidget()
        self.resultTab_2.setObjectName("resultTab_2")
        self.iterationTabWidget_2.addTab(self.resultTab_2, "")
        self.gridLayout.addWidget(self.iterationTabWidget_2, 0, 1, 1,
2)
        self.verticalLayout_4 = QtWidgets.QVBoxLayout()
        self.verticalLayout_4.setObjectName("verticalLayout_4")
        self.gridLayout.addLayout(self.verticalLayout_4, 2, 0, 1, 1)
        MainWindow.setCentralWidget(self.wind)

        self.retranslateUi(MainWindow)
        self.iterationTabWidget_2.setCurrentIndex(1)
        QtCore.QMetaObject.connectSlotsByName(MainWindow)

    def retranslateUi(self, MainWindow):
        _translate = QtCore.QCoreApplication.translate
        MainWindow.setWindowTitle(_translate("MainWindow", "Задача о
рюкзаках"))
        self.backButton.setText(_translate("MainWindow", "Шаг назад"))
        self.forwardButton.setText(_translate("MainWindow", "Шаг
вперед"))
        self.resultButton.setText(_translate("MainWindow",
"Результат"))
        self.ParamsLabel.setText(_translate("MainWindow", "Параметры
работы алгоритма"))
        self.backpackValueLabel.setText(_translate("MainWindow",
"Допустимый вес рюкзака"))

self.backpackValueLE.setPlaceholderText(_translate("MainWindow", "Вес
рюкзака"))
        self.genAmountLabel.setText(_translate("MainWindow",
"Количество поколений"))

self.generationAmountLE.setPlaceholderText(_translate("MainWindow",
"Кол-во поколений"))
        self.entityAmountLabel.setText(_translate("MainWindow",
"Количество особей в поколении"))

self.entityAmountLE.setPlaceholderText(_translate("MainWindow", "Кол-
во особей в поколении"))
        self.probabilityCrossingLabel.setText(_translate("MainWindow",
"Вероятность скрещевания"))

self.probabilityCrossingLabel_2.setText(_translate("MainWindow",
"Вероятность мутации"))
        self.DataLabel.setText(_translate("MainWindow", "Данные"))
        self.randomGenButton.setText(_translate("MainWindow",
"Сгенерировать случайно"))
        self.browseButton.setText(_translate("MainWindow", "Загрузить
из файла"))
        self.inputButton.setText(_translate("MainWindow", "Ввести
вручную"))
```

```
        self.resetDataButton.setText(_translate("MainWindow",
"Сбросить данные"))
        self.startButton.setText(_translate("MainWindow", "Запуск"))

self.iterationTabWidget_2.setTabText(self.iterationTabWidget_2.indexOf
(self.instructionTab_2), _translate("MainWindow", "Инструкция"))
        self.noDataLabel.setText(_translate("MainWindow", "Нет
данных"))
        self.iterationLabel.setText(_translate("MainWindow", "Итерация
номер "))
        self.iterationNumLabel.setText(_translate("MainWindow", "0"))
        item = self.backpackTableWidget.horizontalHeaderItem(0)
        item.setText(_translate("MainWindow", "Вес"))
        item = self.backpackTableWidget.horizontalHeaderItem(1)
        item.setText(_translate("MainWindow", "Стоимость"))
        item = self.backpackTableWidget.horizontalHeaderItem(2)
        item.setText(_translate("MainWindow", "Количество"))
        self.textCurWeightLabel.setText(_translate("MainWindow",
"Текущий вес рюкзака:"))
        self.curWeightLabel.setText(_translate("MainWindow", "-1"))
        self.textFreeSpacLabel.setText(_translate("MainWindow",
"Оставшееся свободное место:"))
        self.freeSpaveLabel.setText(_translate("MainWindow", "-1"))
        self.textCurBPCostLabel.setText(_translate("MainWindow",
"Текущая стоимость рюкзака:"))
        self.curBPCostLabel.setText(_translate("MainWindow", "-1"))
        item = self.backpackTableWidget_2.horizontalHeaderItem(0)
        item.setText(_translate("MainWindow", "Вес"))
        item = self.backpackTableWidget_2.horizontalHeaderItem(1)
        item.setText(_translate("MainWindow", "Стоимость"))
        item = self.backpackTableWidget_2.horizontalHeaderItem(2)
        item.setText(_translate("MainWindow", "Количество"))
        self.textCurWeightLabel_2.setText(_translate("MainWindow",
"Текущий вес рюкзака:"))
        self.curWeightLabel_2.setText(_translate("MainWindow", "-1"))
        self.textFreeSpacLabel_2.setText(_translate("MainWindow",
"Оставшееся свободное место:"))
        self.freeSpaveLabel_2.setText(_translate("MainWindow", "-1"))
        self.textCurBPCostLabel_2.setText(_translate("MainWindow",
"Текущая стоимость рюкзака:"))
        self.curBPCostLabel_2.setText(_translate("MainWindow", "-1"))
        item = self.backpackTableWidget_3.horizontalHeaderItem(0)
        item.setText(_translate("MainWindow", "Вес"))
        item = self.backpackTableWidget_3.horizontalHeaderItem(1)
        item.setText(_translate("MainWindow", "Стоимость"))
        item = self.backpackTableWidget_3.horizontalHeaderItem(2)
        item.setText(_translate("MainWindow", "Количество"))
        self.textCurWeightLabel_3.setText(_translate("MainWindow",
"Текущий вес рюкзака:"))
        self.curWeightLabel_3.setText(_translate("MainWindow", "-1"))
        self.textFreeSpacLabel_3.setText(_translate("MainWindow",
"Оставшееся свободное место:"))
        self.freeSpaveLabel_3.setText(_translate("MainWindow", "-1"))
        self.textCurBPCostLabel_3.setText(_translate("MainWindow",
"Текущая стоимость рюкзака:"))
        self.curBPCostLabel_3.setText(_translate("MainWindow", "-1"))
        self.textSumWeightLabel.setText(_translate("MainWindow",
```

```
"Суммарный вес: "))
        self.sumWeightLabel.setText(_translate("MainWindow", "-1"))
        self.textDeltaWeightLabel.setText(_translate("MainWindow",
"Дельта веса: "))
        self.deltaWeightLabel.setText(_translate("MainWindow", "-1"))
        self.textSumCostLabel.setText(_translate("MainWindow",
"Суммарная стоимость: "))
        self.sumCostLabel.setText(_translate("MainWindow", "-1"))
        self.textDeltaMaxCostLabel.setText(_translate("MainWindow",
"Дельта с макс стоимостью: "))
        self.deltaMaxCostLabel.setText(_translate("MainWindow", "-1"))

self.iterationTabWidget_2.setTabText(self.iterationTabWidget_2.indexOf
(self.iterationTab_2), _translate("MainWindow", "Итерация алгоритма"))

self.iterationTabWidget_2.setTabText(self.iterationTabWidget_2.indexOf
(self.historyOfIterationsTab_2), _translate("MainWindow", "История
итераций"))

self.iterationTabWidget_2.setTabText(self.iterationTabWidget_2.indexOf
(self.resultTab_2), _translate("MainWindow", "Результат"))


if __name__ == "__main__":
    import sys
    app = QtWidgets.QApplication(sys.argv)
    MainWindow = QtWidgets.QMainWindow()
    ui = Ui_MainWindow()
    ui.setupUi(MainWindow)
    MainWindow.show()
    sys.exit(app.exec())
```

Название файла: UIRandGenDialog.py

```
# Form implementation generated from reading ui file 'randGenUI.ui'
#
# Created by: PyQt6 UI code generator 6.7.0
#
# WARNING: Any manual changes made to this file will be lost when
pyuic6 is
# run again.  Do not edit this file unless you know what you are
doing.


from PyQt6 import QtCore, QtGui, QtWidgets


class Ui_randGenDialog(object):
    def setupUi(self, randGenDialog):
        randGenDialog.setObjectName("randGenDialog")
        randGenDialog.resize(400, 152)
        randGenDialog.setMinimumSize(QtCore.QSize(400, 152))
        randGenDialog.setMaximumSize(QtCore.QSize(400, 152))
        self.verticalLayoutWidget =
QtWidgets.QWidget(parent=randGenDialog)
```

```
        self.verticalLayoutWidget.setGeometry(QtCore.QRect(80, 30,
229, 49))

self.verticalLayoutWidget.setObjectName("verticalLayoutWidget")
        self.verticalLayout =
QtWidgets.QVBoxLayout(self.verticalLayoutWidget)
        self.verticalLayout.setContentsMargins(0, 0, 0, 0)
        self.verticalLayout.setObjectName("verticalLayout")
        self.amountLabel =
QtWidgets.QLabel(parent=self.verticalLayoutWidget)

self.amountLabel.setAlignment(QtCore.Qt.AlignmentFlag.AlignCenter)
        self.amountLabel.setObjectName("amountLabel")
        self.verticalLayout.addWidget(self.amountLabel)
        self.horizontalLayout_2 = QtWidgets.QHBoxLayout()
        self.horizontalLayout_2.setObjectName("horizontalLayout_2")
        spacerItem = QtWidgets.QSpacerItem(40, 20,
QtWidgets.QSizePolicy.Policy.Expanding,
QtWidgets.QSizePolicy.Policy.Minimum)
        self.horizontalLayout_2.addItem(spacerItem)
        self.AmountLineEdit =
QtWidgets.QLineEdit(parent=self.verticalLayoutWidget)
        self.AmountLineEdit.setObjectName("AmountLineEdit")
        self.horizontalLayout_2.addWidget(self.AmountLineEdit)
        spacerItem1 = QtWidgets.QSpacerItem(40, 20,
QtWidgets.QSizePolicy.Policy.Expanding,
QtWidgets.QSizePolicy.Policy.Minimum)
        self.horizontalLayout_2.addItem(spacerItem1)
        self.verticalLayout.addLayout(self.horizontalLayout_2)
        self.doneButton = QtWidgets.QPushButton(parent=randGenDialog)
        self.doneButton.setGeometry(QtCore.QRect(230, 110, 158, 32))
        self.doneButton.setObjectName("doneButton")
        self.CancelButton =
QtWidgets.QPushButton(parent=randGenDialog)
        self.CancelButton.setGeometry(QtCore.QRect(20, 110, 121, 32))
        self.CancelButton.setObjectName("CancelButton")

        self.retranslateUi(randGenDialog)
        QtCore.QMetaObject.connectSlotsByName(randGenDialog)

    def retranslateUi(self, randGenDialog):
        _translate = QtCore.QCoreApplication.translate
        randGenDialog.setWindowTitle(_translate("randGenDialog",
"Случайная генерация"))
        self.amountLabel.setText(_translate("randGenDialog", "Введите
количество предметов"))
        self.doneButton.setText(_translate("randGenDialog", "Готово"))
        self.CancelButton.setText(_translate("randGenDialog",
"Назад"))


if __name__ == "__main__":
    import sys
    app = QtWidgets.QApplication(sys.argv)
    randGenDialog = QtWidgets.QDialog()
    ui = Ui_randGenDialog()
    ui.setupUi(randGenDialog)
```

```
randGenDialog.show()
sys.exit(app.exec())
```