

КУРСОВАЯ РАБОТА

Тема: «Прогнозирование вариантов расходов на стоимость компонентов и логистику для производства»

Исполнитель: Кудюров Владислав Владимирович

Москва 2020

## СОДЕРЖАНИЕ:

[**Тема, описание задачи**](#_y8l69auq4j1d) **2**

[**База данных**](#_gfg3mptkksz8) **3**

[**Параметризация данных**](#_ghbytq9yl6un) **3**

[**Архитектура НС**](#_l65kb3ywnrzt) **3**

[**Графическое подтверждение**](#_hd5p6oxq6ks2) **3**

[**Ноутбук**](#_kbfzv6b8xasb) **3**

[**Выводы**](#_s1bgid2xh6is) **3**

[**План дальнейшей работы**](#_b9ew54hqvb7i) **4**

## 

## Тема, описание задачи

Тема диплома: “ Прогнозирование вариантов расходов на стоимость компонентов и логистику для производства”

Одна из основных областей применений генетических алгоритмов – поисковые задачи, которые находят широкое применение в логистике, исследовании операций, искусственном интеллекте и машинном обучении. Поисковый алгоритм решает задачу путем методичного перебора состояний и переходов состояний с целью найти путь из начального состояния в конечное (или целевое). Обычно с каждым переходом состояний ассоциированы некоторые затраты (стоимость) или доходы (выигрыш), а цель поискового алгоритма – найти путь, минимизирующий затраты или максимизирующий выигрыш. Поскольку оптимальный путь – один из многих возможных, такой поиск неразрывно связан с комбинаторной оптимизацией – нахождением оптимального объекта в конечном, но очень большом множестве.

Компания «ВЕЗЁМ» закупает продукцию, управляет центром обработки и исполнения заказов, доставляет посылки заказчикам, располагая всего одним или несколькими автомобилями. Как оптимально выбрать маршрут, чтобы объехать всех заказчиков и вернуться в исходную точку.

Задача курсовой работы:

Пусть дан список локаций и известны расстояния между каждыми двумя локациями. Найти кратчайший путь, проходящий через все локации и возвращающийся в исходную точку.

## База данных

На начальном этапе база данных состоит из двух документов, которых предварительно сериализовали при помощи модуль pickle.

29\_loc.pickle – файл в котором содержаться 29 локаций торговых точек.

29\_dist.pickle – файл в котором содержаться матрица расстояний между 29 локациями.

## Параметризация данных

Извлекаем координаты локаций, вычисляем попарные расстояния между локациями и запоминаем их в матрице расстояний (двумерном массиве).

## Генетический алгоритм

Каждое решение можно оценить, просуммировав расстояния между соседними в последовательности локаций и прибавив еще расстояние между последней и первой локацией. Такой список мы будем использовать для представления хромосомы в генетическом алгоритме. Основные шаги-решения:

* + - 1. Вначале программа создает экземпляр задачи

tsp = TestRoute(‘Тестовый маршрут’)

* + - 1. Определить стратегию приспособления. Мы хотим минимизировать состояние, а это значит, что нужно определить минимизирующий класс Fitness с одной целью, в котором задан один отрицательный вес:

creator.create("FitnessMin", base.Fitness, weights=(-1.0,))

* + - 1. Оптимальное решение задачи представлено такой хромосомой: (0, 27, 5, 11, 8, 25, 2, 28, 4, 20, 1, 19, 9, 3, 14, 17, 13, 16, 21, 10, 18, 24, 6, 22, 7, 26, 15, 12, 23)

приведен код, реализующий хромосому такого вида.

creator.create ("Individual", array.array, typecode='i', fitness=creator.FitnessMin)

toolbox.register("randomOrder", random.sample, range(len(tsp)), len(tsp)) toolbox.register("individualCreator", tools.initIterate, creator.Individual, toolbox.randomOrder) toolbox.register("populationCreator", tools.initRepeat, list, toolbox.individualCreator)

* + - 1. Реализовав хромосому, можем определить функцию вычисления приспособленности.

def tpsDistance(individual):

return tsp.getTotalDistance(individual),

toolbox.register("evaluate", tpsDistance)

* + - 1. Определение генетических операторов, будем использовать турнирный отбор с турниром размера 3

toolbox.register("select", tools.selTournament, tournsize=3)

* + - 1. Но что касается операторов скрещивания и мутации, нужно иметь в виду, что теперь хромосома – не просто список целых чисел, а список индексов, представляющий порядок посещения локаций, поэтому мы не вправе перемешивать части двух списков или произвольно менять индекс. Необходимо использовать специализированные операторы, предназначенные для порождения допустимых списков индексов. Используется реализация этого оператора в каркасе DEAP:

toolbox.register("mate", tools.cxOrdered)

toolbox.register("mutate", tools.mutShuffleIndexes, indpb=1.0/len(tsp))

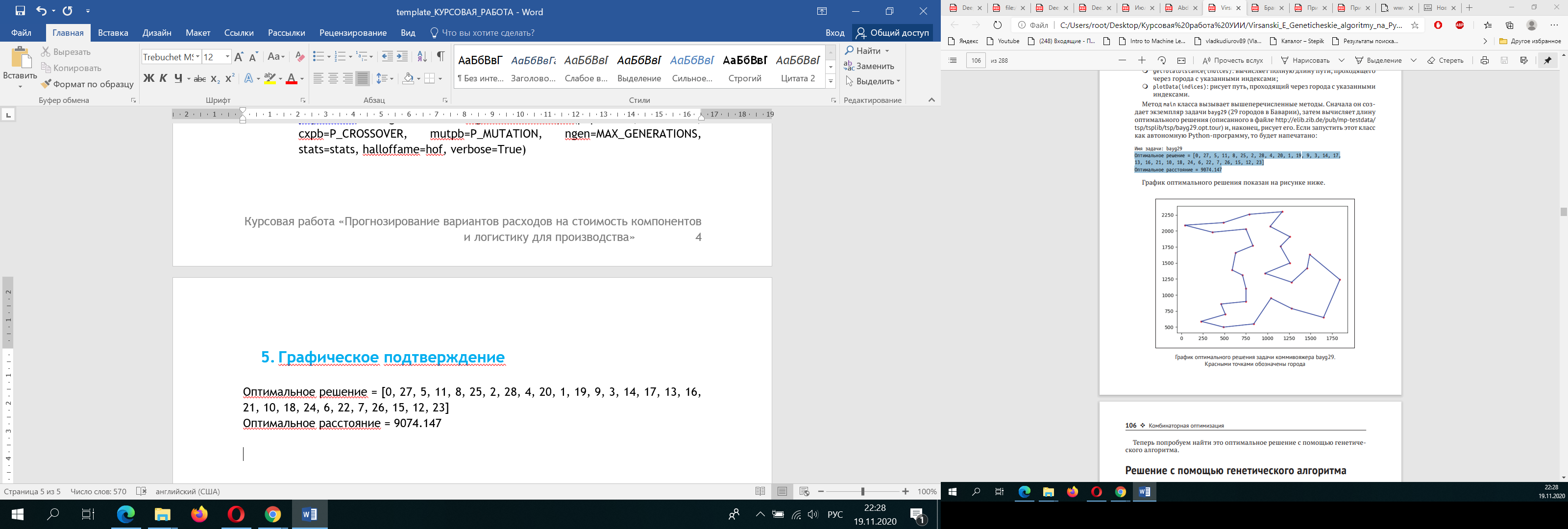
* + - 1. Воспользуемся встроенным в DEAP алгоритмом eaSimple и его объектами статистики и залом славы, которые дают информацию, необходимую для последующего отображения результатов:

population, logbook = algorithms.eaSimple(population, toolbox, cxpb=P\_CROSSOVER, mutpb=P\_MUTATION, ngen=MAX\_GENERATIONS, stats=stats, halloffame=hof, verbose=True)

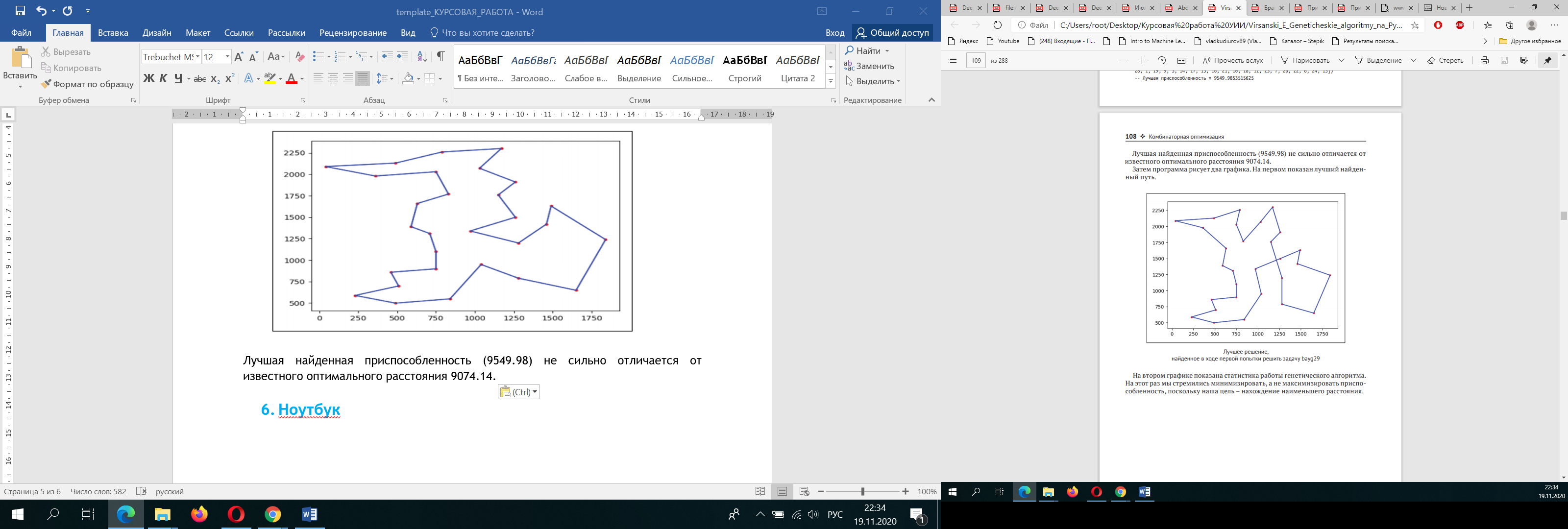
## Графическое подтверждение

Оптимальное решение = [0, 27, 5, 11, 8, 25, 2, 28, 4, 20, 1, 19, 9, 3, 14, 17, 13, 16, 21, 10, 18, 24, 6, 22, 7, 26, 15, 12, 23]

Оптимальное расстояние = 9074.147



Лучшая найденная приспособленность (9549.98) не сильно отличается от известного оптимального расстояния 9074.14.



## Ноутбук

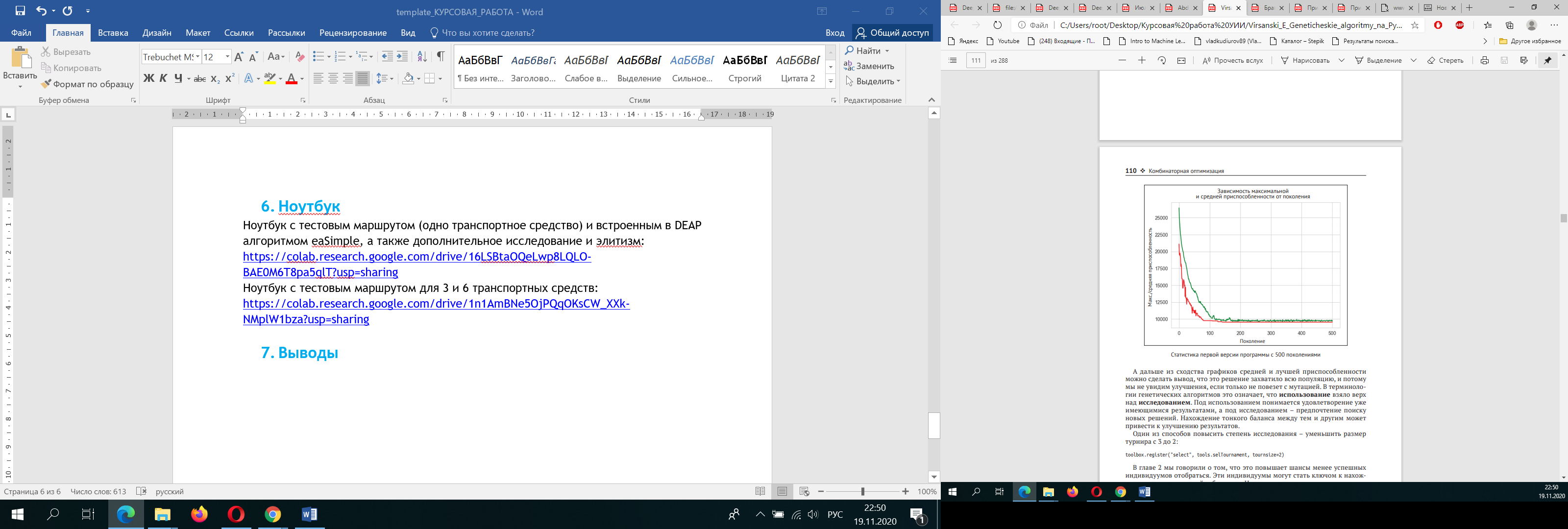
Ноутбук с тестовым маршрутом (одно транспортное средство) и встроенным в DEAP алгоритмом eaSimple, а также дополнительное исследование и элитизм:

<https://colab.research.google.com/drive/16LSBtaOQeLwp8LQLO-BAE0M6T8pa5qlT?usp=sharing>

Ноутбук с тестовым маршрутом для 3 и 6 транспортных средств:

<https://colab.research.google.com/drive/1n1AmBNe5OjPQqOKsCW_XXk-NMplW1bza?usp=sharing>

## Выводы



Из сходства графиков средней и лучшей приспособленности можно сделать вывод, что это решение захватило всю популяцию, и потому мы не увидим улучшения, если только не повезет с мутацией. В терминологии генетических алгоритмов это означает, что использование взяло верх над исследованием. Под использованием понимается удовлетворение уже имеющимися результатами, а под исследованием – предпочтение поиску новых решений. Нахождение тонкого баланса между тем и другим может привести к улучшению результатов.

## План дальнейшей работы

1. Подключение к локальной БД локаций, увеличить число локаций до 100 – 300, транспортных средств до 10 – 30.

2. Модификация и совершенствование алгоритма eaSimple().