Университет ИТМО

Институт прикладных компьютерных наук
Глубокое обучение и генеративный искусственный интеллект

ОТЧЕТ ПО 2-Й ЛАБОРАТОРНОЙ РАБОТЕ
курса
«Эволюционные вычисления»

ВВЕДЕНИЕ В ЭВОЛЮЦИОННЫЕ ВЫЧИСЛЕНИЯ

Студент:
Группа № M4130 *Батурина Ксения Александровна*

Санкт-Петербург 2024

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Для реализации был выбран ЯП **Python**.
Код доступен в репозитории на GitHub: http://www.overleaf.com.

Цель работы:

Получить представление о возможностях применения эволюционных алгоритмов для решения различных классов задач и программных средств для их разработки.

Задачи работы:

1. Скачать проект, содержащий фреймворк Watchmaker.
2. Выполнить работу с примерами: Bits count, Travelling salesman problem, Mona Lisa.
3. Ответить на вопросы:
   - К какому типу по структуре решений относится каждая из рассмотренных задач?
   - Как закодированы решения в задаче коммивояжёра?
   - Что является генотипом, а что фенотипом в задаче воспроизведения картины?

1 BITS COUNT

1.1 Кодирование решений в данной реализации

В данной реализации битовые строки представлены классом `BitString`. Рассмотрим, как решения закодированы в данной структуре данных:

1. Конструкторы:
   - `BitString(int length)`: Создает объект `BitString` заданной длины с массивом целых чисел для хранения битов.
   - `BitString(int length, Random rng)`: Создает объект `BitString` заданной длины и заполняет его случайными битами, используя генератор случайных чисел rng.
   - `BitString(String value)`: Создает объект `BitString` на основе строки `value`, где каждый символ 1 или 0 представляет соответствующий бит.
2. Методы:
   - `getBit(int index)`: Возвращает бит по указанному индексу.
   - `setBit(int index, boolean set)`: Устанавливает бит по указанному индексу в заданное значение.
   - `flipBit(int index)`: Инвертирует бит по указанному индексу.
   - `countSetBits()`: Подсчитывает количество установленных битов (битов со значением 1).
   - `countUnsetBits()`: Подсчитывает количество сброшенных битов (битов со значением 0).
   - `toNumber()`: Преобразует битовую строку в целое число типа `BigInteger`.
   - `swapSubstring(BitString other, int start, int length)`: Меняет подстроку текущей битовой строки с подстрокой другой битовой строки, начиная с позиции `start` и длиной `length`.

Таким образом, решения в данной реализации закодированы в виде массива целых чисел, где каждое целое число представляет группу битов, а методы класса `BitString` обеспечивают манипуляции с этими битами.

1.2 Результаты расчётов

В соответствии с заданием была проведена серия запусков для решения задачи с размерностями 20, 50 и 100. В таблице ниже приведено количество итераций алгоритма для всех запусков и расчет средних значений.

Таблица 1 — Результаты расчёта количества итераций алгоритма от размерности проблемы.

Размерность	Run 1	Run 2	Run 3	Run 4	Run 5	Среднее
20	30	33	19	36	25	28,6
50	1856	5714	1094	2519	4356	3107,8
100	5676697	15910562	129085	24867563	20868152	13490411,8

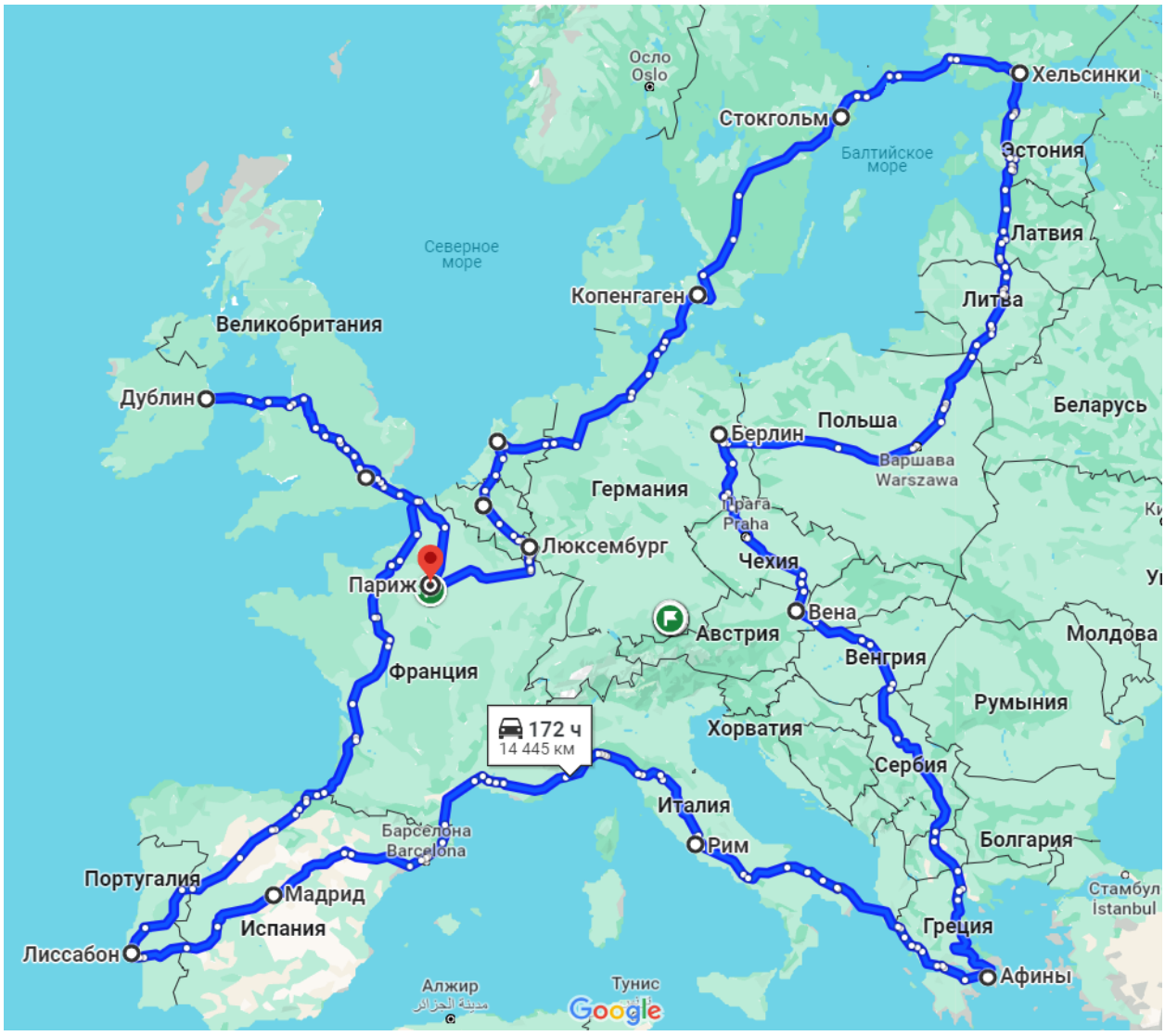
2 TRAVELLING SALESMAN PROBLEM

2.1 Оптимальный маршрут

Найденный оптимальный маршрут: Paris ⟶ Luxembourg ⟶ Brussels ⟶ Amsterdam ⟶ Copenhagen ⟶ Stockholm ⟶ Helsinki ⟶ Berlin ⟶ Vienna ⟶ Athens ⟶ Rome ⟶ Madrid ⟶ Lisbon ⟶ Dublin ⟶ London ⟶ Paris.

Дистанция: 10494,0 км.

Рисунок 1 — Оптимальный маршрут



2.2 Подбор параметров

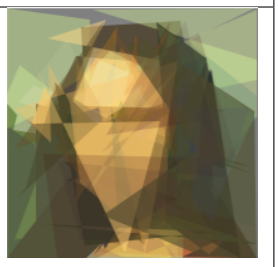
При стандартных параметрах алгоритма (`pop: 300`, `gen: 100`, `elite: 3`, `Truncation Selection (50%)`) оптимальный маршрут, описанный выше, находится за $1{,}86$ секунд.

Были проведены запуски с различными параметрами алгоритма. В частности, удалось достичь стабильного нахождения оптимального пути за $0{,}023$ секунд при параметрах (`pop: 50`, `gen: 100`, `elite: 3`, `Truncation Selection (50%)`).

3 MONA LISA

Качество решения задачи оценивалось субъективно. Говорят, что самая значимая часть этой картины — это улыбка, по этому критерию (а также по значению функции приспособленности) и был выбран «лучший» вариант.

Таблица 2 — Результаты оптимизации подбора полигонов для воспроизведения картины.

Решение	Итерация	Фитнесс	Кол-во полигонов и углов	Рисунок
плохое	70784	226670,45	22 полигонов, 162 углов	
среднее	39578	232315.14	35 полигонов, 238 углов	
хорошее	63149	194167	39 полигонов, 318 углов	

4 ВОПРОСЫ

4.1 К какому типу по структуре решений относится каждая из рассмотренных задач?

- **Bits count.** Тип решений: Бинарные. Решение представляет собой битовую строку (каждый бит может быть 0 или 1), и задача заключается в подсчете количества установленных битов 1 в этой строке.
- **TSP.** Тип решений: Комбинаторные. Решение представляет собой перестановку городов и является комбинаторным объектом.
- **Mona Lisa.** Тип решений: Вещественнозначные. Решение — список полигонов, где параметры (координаты вершин, цвет, прозрачность) могут принимать вещественные значения. Каждый полигон описывается набором вещественных чисел.

4.2 Как закодированы решения в задаче коммивояжёра?

В данной реализации решения закодированы в виде списка строк, представляющих порядок посещения городов в маршруте.

Таким образом, в данной реализации каждая особь представляет маршрут, который является перестановкой городов. Эти маршруты эволюционируют с использованием различных стратегий, представленных интерфейсом TravellingSalesmanStrategy.

4.3 Что является генотипом, а что фенотипом в задаче воспроизведения картины?

Генотип представляет собой список полигонов, каждый из которых описывается своими характеристиками, такими как координаты вершин, цвет и прозрачность.

Фенотип в данной задаче является изображением, созданным на основе генотипа. Полигоны из списка генотипа используются для построения изобра-

жения, которое приближается к целевому изображению (в данном случае, к картине Моны Лизы).