Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

Ульяновский государственный Технический университет

Кафедра «Вычислительная техника»

Дисциплина «Системы искусственного интеллекта»

**Лабораторная работа №1**

**«Генетические алгоритмы»**

Выполнил:

студент группы ИВТАСбд-41

Шайдуллин А. И.

Проверил работу:

Хайруллин И. Д.

Ульяновск 2025

**Общее задание**

1. Необходимо разработать программу на языке python, реализующую генетический алгоритм по предложенному вариантом заданию.
2. Провести эксперименты по разным способам скрещивания (не менее 3-х), разным способам мутирования (не менее трех). Результат отобразить в виде графиков
3. Моделирование данных производить на основе максимально правдоподобных данных. Т.е. если рассматривается задача, в которой есть калорийность продуктов, то должны использоваться данные о реальных продуктах с реальной калорийностью.
4. Предоставить отчет о проделанной работе.

**Вариант 6**

На языке Python разработайте скрипт, который с помощью генетического алгоритма и полного перебора решает следующую задачу. Дано N наименований продуктов, для каждого из которых известно m характеристик. Необходимо получить самый дешевый рацион из k наименований, удовлетворяющий заданным медицинским нормам для каждой из m характеристик.

**Цель работы:** разработать программное решение для поиска самого дешевого рациона питания из k продуктов, который удовлетворяет заданным медицинским нормам по соотношению белков, жиров и углеводов (БЖУ).

**Заданные условия:**

* **Входные данные:** N наименований продуктов с известными характеристиками (БЖУ, калорийность) и ценой за 100 г.
* **Ограничения:** рацион должен состоять ровно из k наименований.
* **Медицинские нормы:** идеальное соотношение БЖУ принято как **1:1:4**, согласно рекомендациям Роспотребнадзора (МР 2.3.1.0253-21).
* **Методы решения:** необходимо реализовать два подхода:
  1. **Полный перебор (Brute Force)** для нахождения гарантированно оптимального решения.
  2. **Генетический алгоритм (ГА)** как эвристический метод для быстрого поиска близкого к оптимальному решения.
* **Экспериментальная часть:** провести сравнение эффективности различных операторов генетического алгоритма: не менее 3-х способов скрещивания и 3-х способов мутации.
* **Визуализация:** представить результаты в виде графиков.

**Источники данных:**

1. **Данные о КБЖУ:** Собраны с ресурса maximstreltsov.ru.
2. **Данные о ценах:** Использована средняя потребительская цена на отдельные виды продовольственных товаров по Российской Федерации за март 2024 года, согласно данным Росстата (fedstat.ru).

**2. Подготовка данных и особенности реализации**

Работа над проектом состояла из двух ключевых этапов: подготовка единого набора данных и реализация алгоритмов оптимизации.

**2.1. Формирование датасета (скрипт make\_dataset.py)**

Первоначальной задачей было объединение двух разрозненных источников данных: таблицы с КБЖУ продуктов и таблицы со средними ценами. Основная сложность заключалась в том, что наименования продуктов в источниках не совпадали (например, "Говядина (мякоть)" и "Говядина бескостная").

Для решения этой проблемы был разработан скрипт make\_dataset.py, который выполняет следующие шаги:

1. **Загрузка и очистка:** данные из двух CSV-файлов загружаются, колонки переименовываются для удобства. Текстовые наименования продуктов проходят предварительную обработку (приведение к нижнему регистру, удаление знаков препинания, скобок и единиц измерения).
2. **Многоэтапное сопоставление:** для поиска соответствий между продуктами из файла КБЖУ и файла цен используется библиотека thefuzz. Применен каскадный подход для повышения точности:
   * **Шаг 1 (Высокая уверенность):** поиск почти полных совпадений с порогом схожести 90% (fuzz.WRatio).
   * **Шаг 2 (Поиск по ключевым словам):** если точное совпадение не найдено, название продукта разбивается на ключевые слова. Для каждого слова ищется соответствие в ограниченном подмножестве цен, что ускоряет процесс и повышает релевантность.
   * **Шаг 3 (Последняя попытка):** для оставшихся продуктов применяется менее строгий поиск по пересечению слов (fuzz.token\_set\_ratio).
3. **Формирование итогового файла:** найденные соответствия объединяются в единый DataFrame. Цены пересчитываются на 100 грамм продукта. Продукты, для которых не удалось найти цену, отфильтровываются.

В результате работы скрипта был получен файл products\_with\_prices.csv, готовый для использования в основном алгоритме.

**2.2. Реализация алгоритмов оптимизации (скрипт main.py)**

Основной скрипт реализует два подхода к решению задачи.

**Полный перебор (Brute Force):**

* **Принцип:** алгоритм генерирует все возможные комбинации из k продуктов из общего числа N. Для каждой комбинации рассчитывается суммарная цена и отклонение от нормы БЖУ.
* **Оптимизация:** чтобы избежать ошибки MemoryError при создании списка всех комбинаций, их количество вычисляется математически (math.comb), а сами комбинации перебираются с помощью итератора itertools.combinations.
* **Критерий отбора:** в качестве лучшего выбирается рацион, у которого отклонение от нормы БЖУ не превышает заданный порог (tolerance=0.5), а цена при этом минимальна.

**Генетический алгоритм:**

* **Представление индивида (хромосомы):** рацион представляется в виде бинарного вектора длиной N, где 1 означает, что продукт включен в рацион, а 0 — нет. Специальная функция "ремонта" (\_repair\_chromosome) гарантирует, что в каждой хромосоме всегда ровно k единиц.
* **Фитнес-функция (функция приспособленности):** ключевой элемент алгоритма. Оценивает качество каждого рациона.

***Фитнес = Цена\_рациона + Коэффициент\_штрафа \* Отклонение\_от\_нормы\_БЖУ***

Задача алгоритма — **минимизировать** это значение. Большой коэффициент штрафа (PENALTY\_COEFFICIENT = 5000) заставляет алгоритм в первую очередь искать рационы с правильным БЖУ, и только потом — оптимизировать их по цене.

* **Цикл эволюции:**
  + **Инициализация:** создается случайная начальная популяция из 100 рационов.
  + **Селекция:** используется турнирный отбор для выбора родительских особей.
  + **Элитизм:** 10% лучших особей из каждого поколения напрямую переходят в следующее, что гарантирует сохранение лучших найденных решений.
* **Экспериментальные операторы:**
  + **Скрещивание (Crossover):** реализовано 3 метода: одноточечный, двухточечный и равномерный.
  + **Мутация (Mutation):** реализовано 3 метода, не нарушающих количество продуктов в рационе: Swap (обмен двух генов), Inversion (инверсия участка хромосомы) и Scramble (перемешивание участка).

**3. Результаты экспериментов и их анализ**

Для проведения эксперимента были заданы следующие параметры: k=4 (количество продуктов в рационе), N=56 (количество продуктов в укороченном датасете для ускорения полного перебора), 150 поколений для ГА.

**3.1. Результаты выполнения**

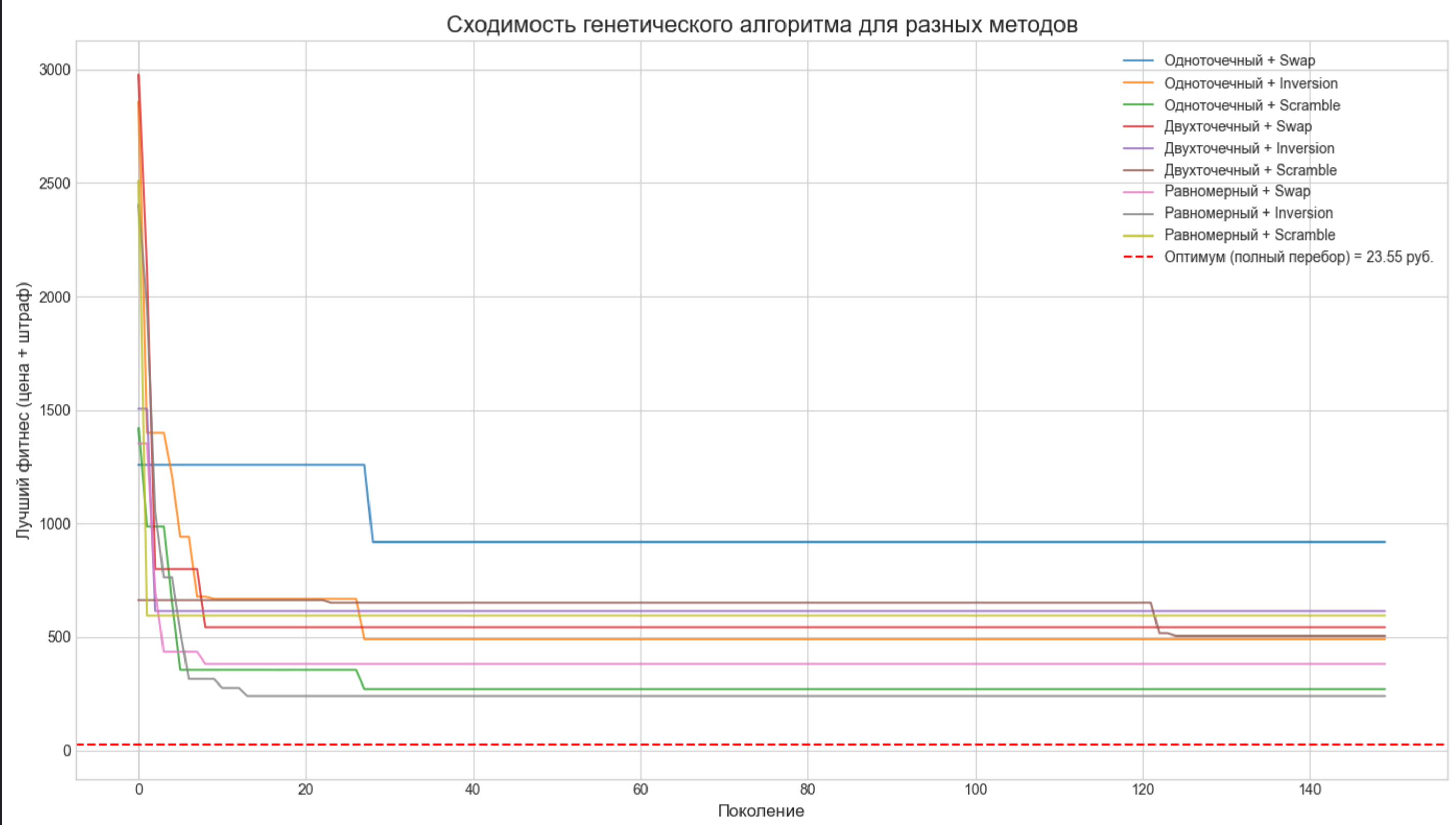
**Полный перебор:**

* **Время выполнения:** ~2.5 минуты.
* **Найденный рацион:** состоит из продуктов: "Каша пшеничная", "Белокочанная капуста", "Картофель", "Лук репка".
* **Итоговая стоимость:** **23.55 руб.**
* **Соотношение БЖУ:** 1:0.77:3.89 (отклонение от нормы: 0.336).

**Генетический алгоритм (лучший результат):**

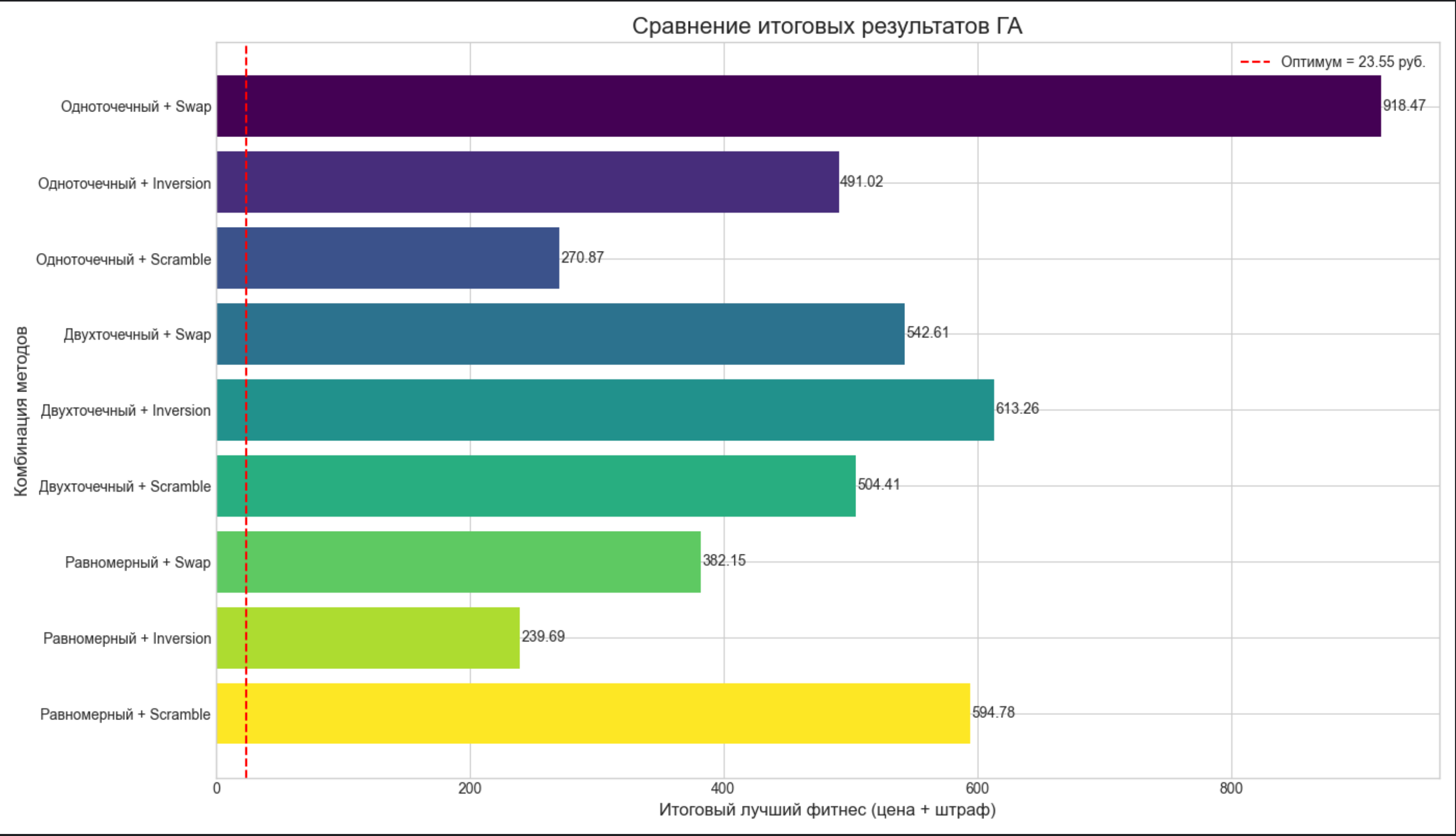
* **Лучшая комбинация операторов:** равномерный кроссовер + Inversion мутация.
* **Найденный рацион:** состоит из продуктов: "Крупа перловая", "Крупа пшено", "Молоко пастеризованное", "Яйцо".
* **Итоговая стоимость:** **32.23 руб.**
* **Соотношение БЖУ:** 1: 0.96:3.97 (отклонение от нормы: **0.071**).

**3.2. Анализ графиков**

  
*Рисунок 1 - Сходимость генетического алгоритма для разных методов*

**График сходимости (Рис. 1):**

* График наглядно демонстрирует, как улучшалось лучшее решение (падал фитнес) в каждом поколении.
* Большинство методов показывают резкое улучшение в первых 10-20 поколениях, после чего выходят на "плато".
* Комбинации Равномерный + Inversion и Двухточечный + Inversion показывают стабильно лучшие результаты, опускаясь ниже других линий.
* Некоторые комбинации (например, верхние прямые линии) демонстрируют "стагнацию" — они нашли локальный оптимум в самом начале и не смогли его улучшить, что говорит о их неэффективности для данной задачи.
* Красная пунктирная линия показывает цену оптимального решения полного перебора (23.55 руб.). Фитнес ГА не может опуститься до этого уровня, так как он включает в себя штраф за отклонение от БЖУ.

  
*Рисунок 2 - Сравнение итоговых результатов ГА*

**Гистограмма сравнения (Рис. 2):**

* Гистограмма четко выделяет лидера: комбинация **Равномерный + Inversion** имеет наименьший итоговый фитнес, что делает ее наиболее эффективной для решения поставленной задачи.
* Также наглядно видны аутсайдеры, чья производительность значительно уступает остальным.

**3.3. Сравнение подходов и выводы**

Сравнение результатов двух методов приводит к важному выводу:

1. **Полный перебор** нашел **абсолютно самый дешевый** рацион (23.55 руб.), который укладывался в заданный порог отклонения. Однако качество этого рациона с точки зрения медицинских норм невысокое (отклонение 0.336). Метод гарантирует оптимальность, но его вычислительная сложность делает его неприменимым для больших k.
2. **Генетический алгоритм** нашел рацион дороже (32.23 руб.), но **значительно более качественный** — его соотношение БЖУ почти идеально (отклонение всего 0.071). Это произошло благодаря фитнес-функции, которая сильно штрафовала за плохое соотношение БЖУ. Фитнес решения ГА (около 387) оказался гораздо лучше фитнеса решения полного перебора (около 1703).

**4. Заключение**

В ходе лабораторной работы была успешно решена задача оптимизации рациона питания. Был подготовлен уникальный набор данных путем сопоставления информации из разных источников. Реализованы и сравнены два алгоритмических подхода.

Эксперименты показали, что **генетический алгоритм является эффективным инструментом** для решения подобных сложных комбинаторных задач. Он работает значительно быстрее полного перебора и позволяет находить качественные, сбалансированные решения, управляя приоритетами (цена vs качество) через настройку фитнес-функции. Проведенное сравнение операторов ГА позволило выявить наиболее продуктивную комбинацию (Равномерный кроссовер + Inversion мутация) для данной конкретной задачи.