Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное

образовательное учреждение высшего образования

«Ульяновский государственный технический университет»

Кафедра «Информационные системы и технологии»

**Методы и технологии интеллектуальной обработки и анализа данных**

**Лабораторная работа №1**

**«Исследование генетического алгоритма»**

Выполнил:

Студент группы ИСТмд-11

Зейнетдинов М.М.

Проверил:

к.т.н, доцент кафедры «ИВК»

Шишкин В.В.

г. Ульяновск

2025

# Формулировка проблемы

Поиск глобального максимума функции с помощью метода полного перебора (линейного поиска) требует вычисления значения во всех точках, что требует большого времени выполнения при большом диапазоне данных.

В данных ситуациях альтернативой являются методы псевдостохастической оптимизации, в частности генетический алгоритм. Однако особенностью его является то, что взамен своему быстрому времени выполнения он не гарантирует точного глобального максимума, поэтому следует проблемой является подбор параметров, которые смогут достичь лучшего результата по скорости и по точности.

# Гипотеза

При правильно выбранных параметрах (тип скрещивания, уровень мутации, размер популяции, элитаризм и т. П.) генетический алгоритм сможет с высокой вероятностью приближённо находить глобальный максимум быстрее, чем линейный поиск, сохраняя при этом приемлемую точность (ошибка в несколько процентов от истинного максимума).

# План исследования

1. Реализация базового ГА: одноточечное скрещивание, побитовая мутация, турни­рный отбор, элитаризм.
2. Добавление и сравнение скрещивания uniform от one point
3. Создание сетки параметров: для population\_size, max\_generations, cxpb, mutpb, mut\_per\_bit и crossover задать наборы значений и провести полный перебор комбинаций.
4. Сравнение с эталоном: для каждой конфигурации проводить 3 повторных запуска ГА (разные seed), а также один запуск линейного поиска.
5. Сбор статистики: для каждой конфигурации фиксировать — среднюю и СКО финальной лучшей приспособленности ГА, среднее и минимальное время исполнения, абсолютные и относительные ошибки относительно линейного максимума, и истории сходимости по поколениям.
6. Визуализация: три ключевых графика — сравнение времени, сходимость по поколениям и динамика ошибки.

# План эксперимента

Для того чтобы провести эксперимент были подобраны параметры, которые в последствии будут скомбинированы, и на этих значениях получится наглядное отличие всех параметров друг от друга.

Битовая длина для числа была выбрана 15, поскольку более меньшие значения алгоритм линейного поиска находил решения довольно быстро.

Для сравнения алгоритма в разных ситуациях было создано 2 режима: функция (математическая функция) и распределение (распределение Гаусса). Для них было подобрано x\_min = -1000, x\_max = 1000, длина массива 1000000.

На каждую конфигурацию повторно проводится 5 экспериментов с тем же подбором параметров, но с различным seed, для того чтобы определять среднее значения, а не случайные успешные или плохие варианты.

Сетка параметров, по которой будут проведены различные комбинации параметров должна включать в себя сильно различающие параметры:

Листинг 1. Сетка параметров

grid = {  
 "bit\_length": [BIT\_LENGTH],  
 "population\_size": [500,1000],  
 "max\_generations": [50,100],  
 "cxpb": [0.1, 0.7],  
 "mutpb": [round(0.2 / BIT\_LENGTH, 3), 0.2],  
 "crossover": ["onepoint", "uniform"],  
 "mutation": ["flip"],  
 "mut\_per\_bit": [True, False],  
 "elite\_count": [2, 5]  
}

Режим отчёта времени: замеры времени включают всю работу алгоритма (инициализация, все поколения и вычисления fitness).

Метрики: best\_fitness (GA и linear), время работы, mean/std по повторам, абсолютная ошибка GA\_best - linear\_best, относительная ошибка.

# Реализация эксперимента

В реализации эксперимента используются три основных класса: GeneticAlgorithm, LinearSearch и ExperimentRunner.

Оценка приспособленности (fitness evaluation) реализуется в двух режимах. В режиме distribution используется заранее вычисленный массив значений функции, и индекс текущей особи просто используется для быстрого доступа к соответствующему элементу массива. Это позволяет существенно ускорить вычисления.

В режиме function индекс особи преобразуется в действительное значение аргумента функции в заданном диапазоне. После преобразования вычисляется значение функции приспособленности через стандартную функцию eval с передачей необходимого контекста, включая библиотеку математических функций и текущее значение аргумента. Этот режим более гибкий, но требует аккуратного обращения с областью определения функции, особенно если она включает логарифмы или другие операции с ограниченной областью допустимых значений.

В реализации генетического алгоритма используется элитаризм, при котором сохраняется заданное количество лучших особей на каждом поколении. Для подбора родителей применяется турнирный отбор, обеспечивающий баланс между случайностью и качеством отбора.

Для рекомбинации особей могут использоваться два типа скрещивания: одноточечное скрещивание и униформное скрещивание. Мутация реализована либо как побитовый flip, либо как изменение одного конкретного бита. Такой подход обеспечивает разнообразие популяции и снижает риск преждевременной сходимости алгоритма к локальным экстремумам.

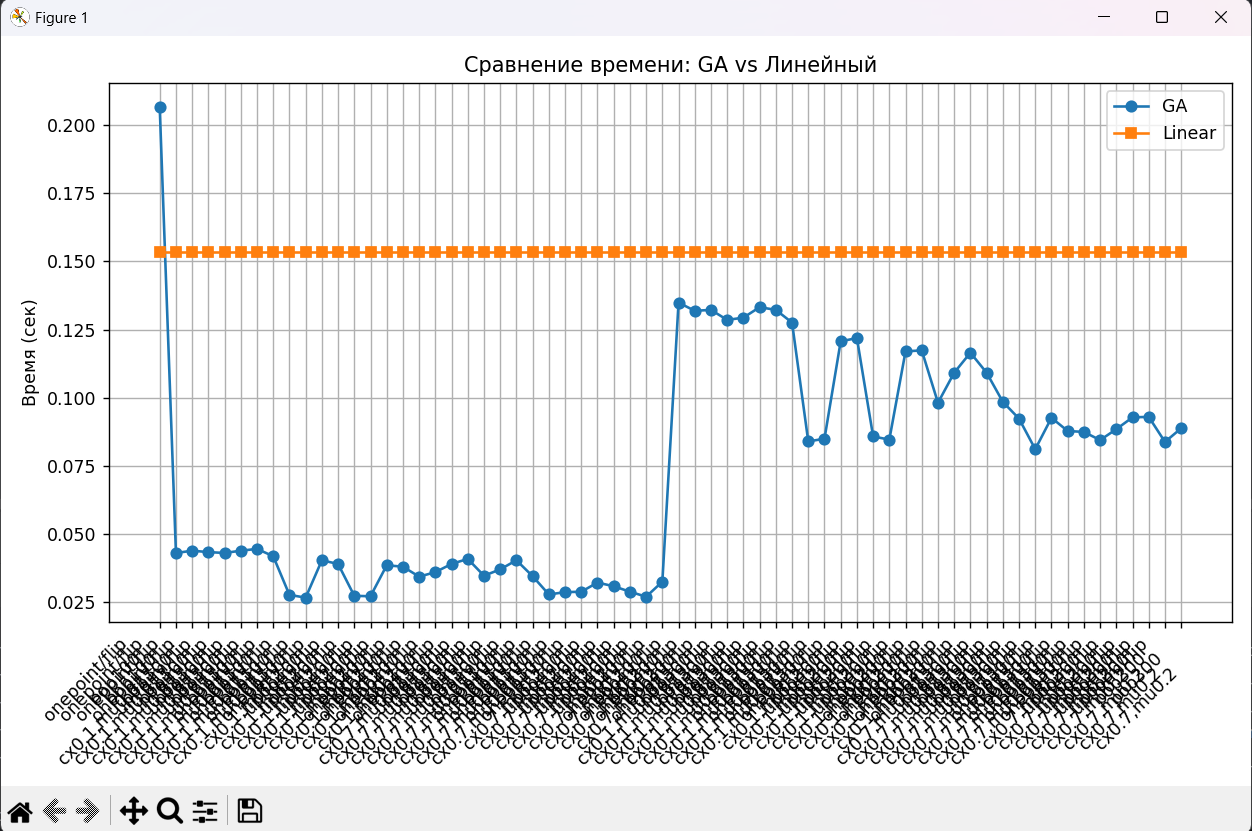


Рисунок 1. График сравнения времени выполнения при случайно заполненном массиве.

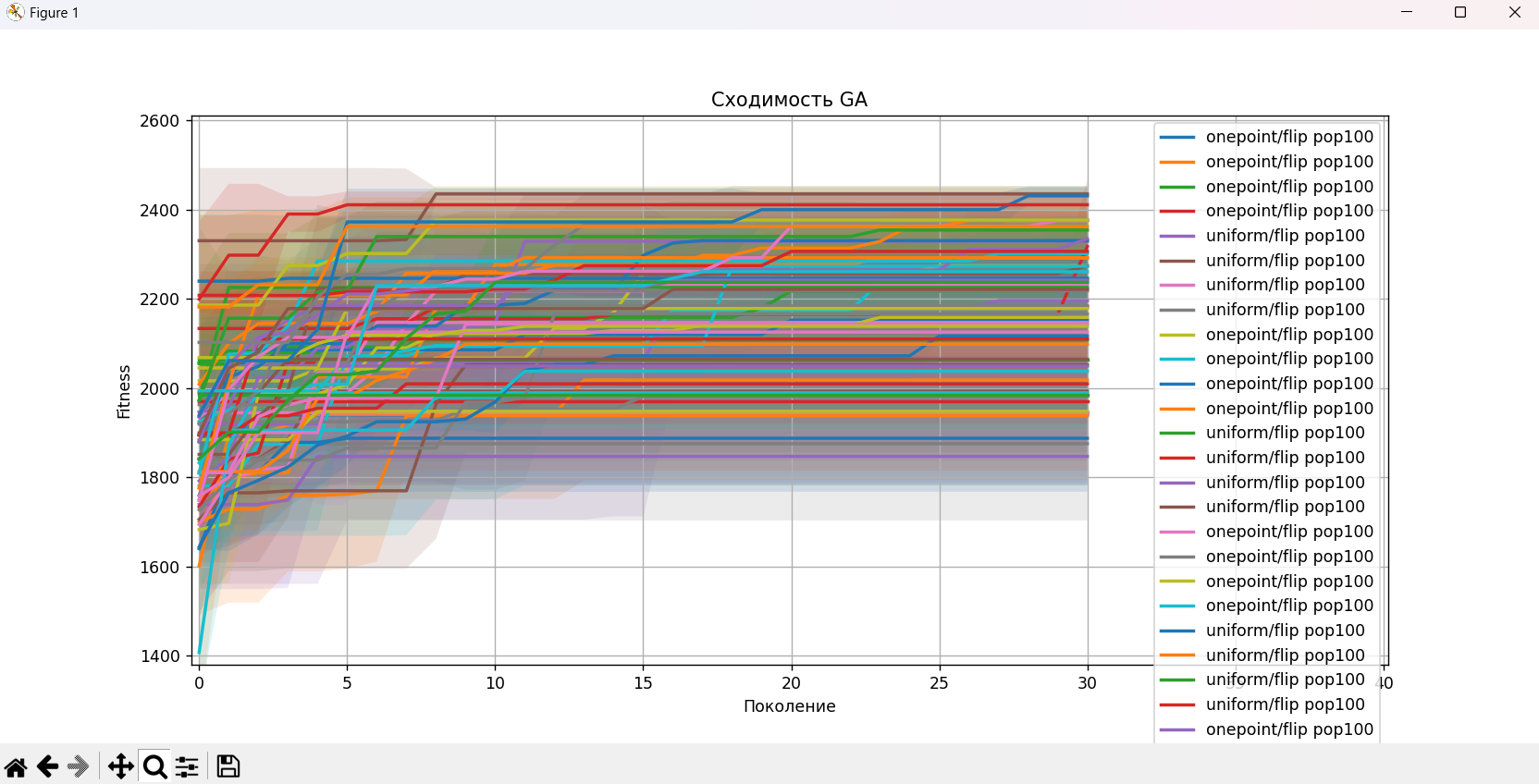


Рисунок 2. График сравнения сходимости ГА от конфигураций

Листинг 2. Результаты эксперимента с распределением

Линейный поиск: лучшая приспособленность = 2445.348376, время = 0.1533 с  
  
Список конфигураций по макс значению GA

1. Конфигурация 14: crossover=uniform, mutation=flip, pop=100, cxpb=0.1, mutpb=0.2, mut\_per\_bit=True -> best=2445.348376, GA accuracy=1.000, mean\_time=0.0272 с

2. Конфигурация 25: crossover=onepoint, mutation=flip, pop=100, cxpb=0.7, mutpb=0.2, mut\_per\_bit=True -> best=2445.348376, GA accuracy=1.000, mean\_time=0.0279 с

3. Конфигурация 26: crossover=onepoint, mutation=flip, pop=100, cxpb=0.7, mutpb=0.2, mut\_per\_bit=True -> best=2445.348376, GA accuracy=1.000, mean\_time=0.0287 с

4. Конфигурация 27: crossover=onepoint, mutation=flip, pop=100, cxpb=0.7, mutpb=0.2, mut\_per\_bit=False -> best=2445.348376, GA accuracy=1.000, mean\_time=0.0289 с

5. Конфигурация 41: crossover=onepoint, mutation=flip, pop=300, cxpb=0.1, mutpb=0.2, mut\_per\_bit=True -> best=2445.348376, GA accuracy=1.000, mean\_time=0.0840 с

Список конфигураций по среднему значению GA

1. Конфигурация 46: crossover=uniform, mutation=flip, pop=300, cxpb=0.1, mutpb=0.2, mut\_per\_bit=True -> mean=2435.020409, GA accuracy=0.996, mean\_time=0.0846 с

2. Конфигурация 61: crossover=uniform, mutation=flip, pop=300, cxpb=0.7, mutpb=0.2, mut\_per\_bit=True -> mean=2430.805181, GA accuracy=0.994, mean\_time=0.0928 с

3. Конфигурация 64: crossover=uniform, mutation=flip, pop=300, cxpb=0.7, mutpb=0.2, mut\_per\_bit=False -> mean=2410.768944, GA accuracy=0.986, mean\_time=0.0888 с

4. Конфигурация 58: crossover=onepoint, mutation=flip, pop=300, cxpb=0.7, mutpb=0.2, mut\_per\_bit=True -> mean=2376.614287, GA accuracy=0.972, mean\_time=0.0875 с

5. Конфигурация 59: crossover=onepoint, mutation=flip, pop=300, cxpb=0.7, mutpb=0.2, mut\_per\_bit=False -> mean=2376.614287, GA accuracy=0.972, mean\_time=0.0845 с

Результаты эксперимента показывают, что выбор параметров генетического алгоритма сильно влияет на качество решения и стабильность результатов. В конфигурациях с низкой вероятностью мутации и малым коэффициентом кроссовера среднее значение приспособленности обычно ниже, а разброс результатов большой. Это говорит о том, что алгоритм в таких настройках часто застревает в локальных оптимумах и не исследует пространство решений достаточно глубоко. Линейный поиск, напротив, показывает стабильный результат, но время его работы больше, чем у большинства запусков ГА.

Когда мутация применяется по отдельным битам, а не целиком, наблюдается рост средней приспособленности в большинстве случаев. Такой подход позволяет более тонко корректировать решения, что особенно важно в случайных массивах с гауссовым распределением. В тех случаях, когда mut\_per\_bit отключен, результаты становятся более разрозненными, а стандартное отклонение увеличивается, а средние значения приспособленности могут как улучшаться, так и ухудшаться.

Тип кроссовера тоже играет роль. Одноточечный кроссовер в сочетании с низким коэффициентом crossover обычно даёт стабильные, но не самые высокие результаты. Равномерный кроссовер увеличивает вариативность потомков, что может повышать шанс найти лучшее решение, но иногда увеличивает разброс результатов. При больших значениях коэффициента crossover алгоритм быстрее смешивает гены и чаще достигает более высоких значений приспособленности, хотя стабильность чуть падает.

Увеличение вероятности мутации до 0.2 заметно улучшает средние результаты для некоторых конфигураций, особенно если мутация применяется по битам. Высокая мутация помогает выходить из локальных оптимумов, но если mut\_per\_bit отключен, эффект становится менее предсказуемым, поскольку иногда разброс сильно увеличивается, а среднее качество снижается.

Размер популяции также влияет на результаты. Увеличение с 100 до 200 индивидов незначительно увеличивает среднюю приспособленность, но при этом время работы алгоритма остаётся очень малым, благодаря случайной природе данных. Большая популяция даёт больше разнообразия, что помогает лучше исследовать пространство решений и снижает вероятность сильных провалов в отдельных запусках.

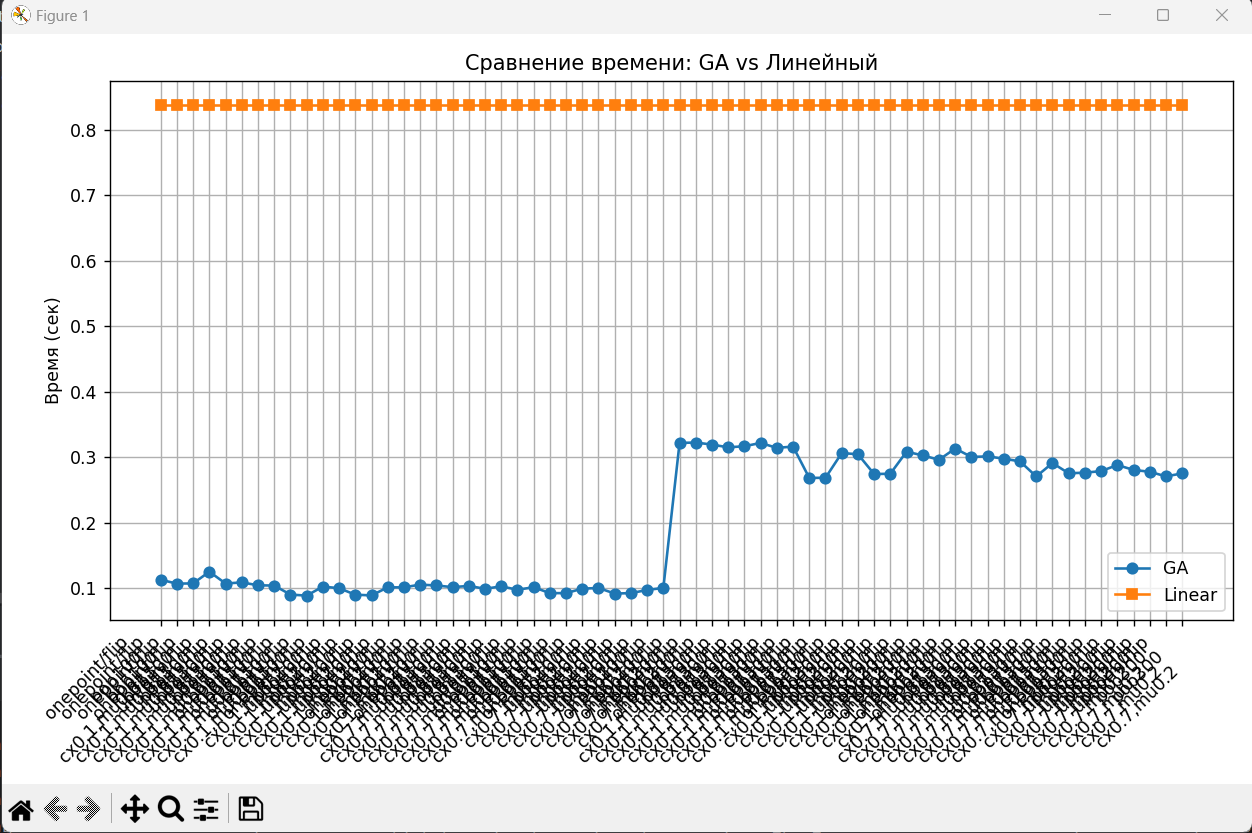


Рисунок 3. График сравнения времени выполнения при мат функции

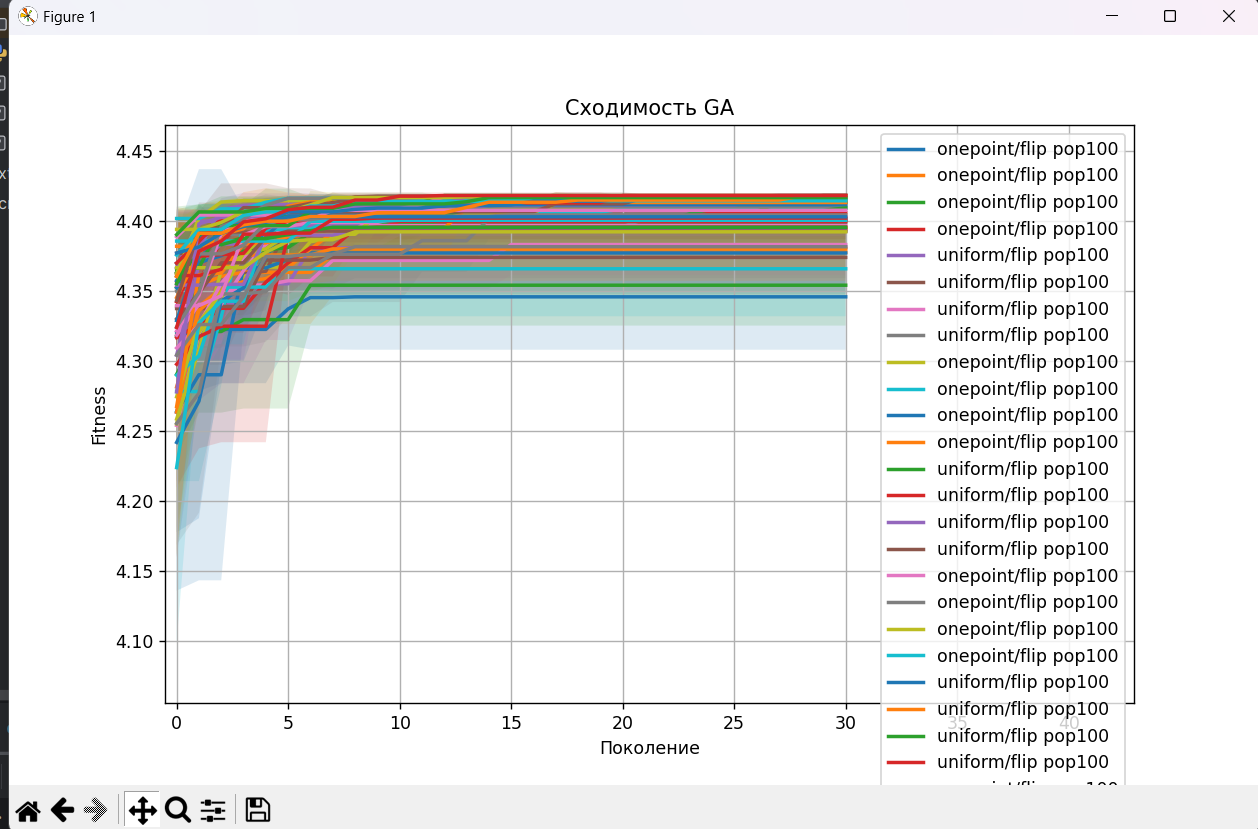


Рисунок 4. График сравнения сходимости ГА от конфигураций

Листинг 3. Результаты эксперимента с мат функцией

Линейный поиск: лучшая приспособленность = 4.418319, время = 0.8378 с  
  
Список конфигураций по макс значению GA

1. Конфигурация 2: crossover=onepoint, mutation=flip, pop=100, cxpb=0.1, mutpb=0.013, mut\_per\_bit=True -> best=4.418319, GA accuracy=1.000, mean\_time=0.1067 с

2. Конфигурация 9: crossover=onepoint, mutation=flip, pop=100, cxpb=0.1, mutpb=0.2, mut\_per\_bit=True -> best=4.418319, GA accuracy=1.000, mean\_time=0.0900 с

3. Конфигурация 13: crossover=uniform, mutation=flip, pop=100, cxpb=0.1, mutpb=0.2, mut\_per\_bit=True -> best=4.418319, GA accuracy=1.000, mean\_time=0.0898 с

4. Конфигурация 21: crossover=uniform, mutation=flip, pop=100, cxpb=0.7, mutpb=0.013, mut\_per\_bit=True -> best=4.418319, GA accuracy=1.000, mean\_time=0.0988 с

5. Конфигурация 26: crossover=onepoint, mutation=flip, pop=100, cxpb=0.7, mutpb=0.2, mut\_per\_bit=True -> best=4.418319, GA accuracy=1.000, mean\_time=0.0921 с

Список конфигураций по среднему значению GA

1. Конфигурация 45: crossover=uniform, mutation=flip, pop=300, cxpb=0.1, mutpb=0.2, mut\_per\_bit=True -> mean=4.418319, GA accuracy=1.000, mean\_time=0.2741 с

2. Конфигурация 46: crossover=uniform, mutation=flip, pop=300, cxpb=0.1, mutpb=0.2, mut\_per\_bit=True -> mean=4.418319, GA accuracy=1.000, mean\_time=0.2746 с

3. Конфигурация 53: crossover=uniform, mutation=flip, pop=300, cxpb=0.7, mutpb=0.013, mut\_per\_bit=True -> mean=4.418319, GA accuracy=1.000, mean\_time=0.2976 с

4. Конфигурация 54: crossover=uniform, mutation=flip, pop=300, cxpb=0.7, mutpb=0.013, mut\_per\_bit=True -> mean=4.418319, GA accuracy=1.000, mean\_time=0.2934 с

5. Конфигурация 64: crossover=uniform, mutation=flip, pop=300, cxpb=0.7, mutpb=0.2, mut\_per\_bit=False -> mean=4.418319, GA accuracy=1.000, mean\_time=0.2751 с

Результаты показывают, что на конкретной функции поведение генетического алгоритма отличается от того, что наблюдается на случайно заполненных массивах. В первую очередь это связано с тем, что функция имеет определённую структуру и гладкость, которая задаёт более предсказуемую карту приспособленности. В отличие от случайного массива, где значения распределены хаотично и локальные оптимумы встречаются практически везде, функция часто содержит выраженные глобальные и локальные экстремумы, что изменяет динамику поиска.

Из-за этого алгоритм быстрее находит хорошие решения даже при относительно низкой вероятности мутации и небольшом коэффициенте кроссовера. Провалы в приспособленности менее вероятны, поскольку пространство решений организовано закономерно. Одновременно с этим эффект мутации становится более предсказуемым: мутация по битам аккуратно корректирует решение и помогает преодолевать локальные экстремумы, а при отключении битовой мутации алгоритм может застревать в локальных максимумах.

Тип кроссовера также проявляется иначе. Одноточечный кроссовер на функции даёт стабильный рост приспособленности, так как части решения, уже близкие к оптимуму, комбинируются достаточно эффективно. Равномерный кроссовер увеличивает вариативность потомков, что иногда не даёт преимущества на гладкой функции, так как случайные перестановки могут отодвигать решение от максимума.

# Вывод

Эффективность генетического алгоритма сильно зависит от выбора параметров, и их влияние проявляется по-разному на случайных массивах и на структурированных функциях. Наиболее стабильные и качественные результаты достигаются при следующих условиях:

Вероятность мутации: умеренные или высокие значения (≈0.1–0.2) помогают алгоритму избегать локальных оптимумов и более полно исследовать пространство решений. При слишком низкой мутации алгоритм застревает, особенно на хаотичных данных.

Мутация по битам: улучшает точность корректировки решений и повышает среднее качество, особенно на данных с распределением, приближенным к нормальному, и на гладких функциях. Отключение этой опции делает результаты менее предсказуемыми и увеличивает разброс.

Коэффициент кроссовера: большие значения ускоряют смешивание генов и чаще приводят к высоким значениям приспособленности, но могут снижать стабильность. Малые значения дают более предсказуемый рост приспособленности, но поиск глобального максимума замедляется.

Тип кроссовера: одноточечный кроссовер обеспечивает стабильный, но не всегда оптимальный рост приспособленности; равномерный кроссовер увеличивает вариативность и шанс найти лучшие решения, но повышает разброс.

Размер популяции: увеличение числа индивидов с 100 до 200 слегка улучшает среднюю приспособленность и снижает вероятность провалов в отдельных запусках, при этом время работы остаётся умеренным.

Генетический алгоритм работает медленнее на случайно заполненных массивах по сравнению с гладкой функцией из-за большей хаотичности пространства решений. В случайных массивах нет явной структуры или закономерности, которая могла бы направлять алгоритм к улучшению приспособленности, поэтому большинство мутаций и скрещиваний дают случайные изменения, часто не приближающие к оптимуму. Это увеличивает количество итераций, необходимых для достижения хорошего результата, и приводит к большему разбросу приспособленности между поколениями.