МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение высшего образования
«Тольяттинский государственный университет»

Кафедра «Прикладная математика и информатика»
(наименование)

01.04.02 Прикладная математика и информатика
(код и наименование направления подготовки)

Математическое моделирование
(направленность (профиль))

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА (МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ)

на тему: «Разработка алгоритмического подхода в задаче оптимизации с использованием генетического алгоритма»

Обучающийся Д.В. Ханипов
(Инициалы Фамилия) (личная подпись)

Научный руководитель к.т.н., доцент, Н.А. Сосина
(ученая степень (при наличии), ученое звание (при наличии), Инициалы Фамилия)

Тольятти 2024

Оглавление

Введение

Задачи оптимизации играют важную роль в различных областях, таких как инженерия, экономика, логистика, технические науки и другие. Разработка эффективных методов оптимизации для решения сложных задач является ключевым аспектом в современной науке и инженерии.

Магистерская диссертация посвящена разработке алгоритмического подхода на основе генетического алгоритма к задаче оптимизации.

Актуальность диссертации состит в том, что в настоящее время генетические алгоритмы, являются мощным инструментом решения сложных задач оптимизации и поиска наилучших решений из множества возможных вариантов. Благодаря своей способности к адаптации, изменению, а также исследованию большого числа вариантов решений, генетические алгоритмы представляются привлекательным инструментом для эффективной оптимизации разнообразных задач на основе процессов эволюции в природе.

Цель магистерской диссертации заключается в исследование потенциала генетических алгоритмов и их применении в решении задач оптимизации.

Задачи магистерской диссертации:

1) анализ математических методов оптимизации;
2) анализ и выбор программных средств для решения задачи оптимизации;
3) разработка алгоритмического подхода для решения поставленной задачи оптимизации с помощью генетического аогоритма;
4) решение тестируемой задачи с помощью генетического алгоритма.

Диссертационная работа состоит из введения, трех глав, заключения и списка используемой литературы и используемых источников.

Глава 1 Математическая теория оптимизации

1.1 Общая постановка задачи оптимизации

Оптимизация - это процесс нахождения наилучшего, наиболее эффективного или оптимального решения для определенной задачи или проблемы. Она может включать в себя максимизацию выгоды, минимизацию издержек, оптимизацию использования ресурсов или достижение оптимального баланса между различными факторами.

Термин "оптимум" и теория оптимизации имеют свое происхождение в различных сферах знаний - от философии до математики. Готфрид В. Лейбниц, в своих философских трудах, рассматривал существующий мир как возможный "оптимум", по сути называя его наилучшим из всех возможных миров. Этот концепт также стал связан с религиозными и мифологическими представлениями, поскольку понятие "оптимум" ассоциировалось с богиней Опы в древнеиталийской мифологии. «На практике оказывается, что в большинстве случаев понятие «наилучший» может быть выражено количественными критериями – минимум затрат, минимум времени, максимум прибыли и т.д. Поэтому возможна постановка математических задач отыскания оптимального (optimum – наилучший) результата, так как принципиальных различий в отыскании наименьшего или наибольшего значения нет. Задачи на отыскание оптимального решения называются задачами оптимизации»[16].

Важную роль в развитии теории оптимизации сыграли математические открытия и разработки в различные исторические периоды. Пьер Ферма в 17 веке установил закономерность, согласно которой при приближении к точкам максимума и минимума скорость функции падает до нуля. Это стало важным шагом в понимании математических свойств оптимизации.

В 18 веке Даниил Бернулли, Леонард Эйлер, Жозеф Л. Лагранж, Карл Вейерштрасс и другие математики начали работать в области вариационного исчисления и изучения задач оптимизации и экстремумов.

История теории оптимизации показывает переход от идей, связанных с философским и религиозным контекстом, к развитию математических методов, которые впоследствии стали фундаментальными для различных областей науки и практики, включая прикладную математику, экономику, инженерные науки, логистику и многие другие сферы.

С появлением электронных вычислительных машин в середине 20 века возможности применения теории оптимизации значительно расширились. Это привело к развитию эффективных численных методов, таких как симплекс-метод, метод динамического программирования и другие, которые нашли свое применение в различных областях науки и инженерии.

Математика дает удобные и плодотворные способы описания самых разнообразных явлений реального мира и, тем самым, выполняет в этом смысле функцию языка.

«Оптимальный результат, как правило, находится не сразу, а в результате процесса, называемого процессом оптимизации. Применяемые в процессе оптимизации методы получили название методов оптимизации» [26].

«Для того, чтобы использовать результаты и вычислительные процедуры теории оптимизации на практике, необходимо, прежде всего, сформулировать рассматриваемую задачу на математическом языке, т.е. построить математическую модель объекта оптимизации. Математическая модель – это более или менее полное математическое описание исследуемого процесса или явления» [25].

«В большинстве реальных ситуаций дать исчерпывающее математическое представление оптимизируемой системы с учетом всех взаимосвязей ее частей, взаимодействий с внешним миром, всех целей ее функционирования бывает затруднительно или невозможно. Поэтому при построении математической модели необходимо, как правило, выделять и учитывать в дальнейшем только наиболее важные, существенные стороны исследуемого объекта с тем, чтобы было возможным его математическое

описание, а также последующее решение поставленной математической задачи. При этом неучтенные в математической модели факторы не должны существенно влиять на окончательный результат оптимизации. Таким образом, математическое моделирование является сложной и ответственной творческой задачей, требующей от исследователя глубоких знаний в соответствующей области, практического опыта, интуиции и критического анализа получаемых результатов» [20].

«Для применения теории оптимизации к решению конкретных задачи нужно выполнить определённую последовательность действий, которая называется постановкой задачи оптимизации. Перечислим этапы задачи оптимизации.

1. Установление границ подлежащей оптимизации системы. Система – некая изолированная часть внешнего мира. Границы системы задают пределы, отделяющие её от внешнего мира. При этом предполагается, что взаимосвязи с внешним миром зафиксированы. Первоначальный выбор границ системы может оказаться слишком жёстким. Для получения адекватного решения нужно включить в систему дополнительные подсистемы, однако это ведёт к увеличению размерности задачи. Следует стремиться к представлению системы в виде изолированных подсистем, которые можно рассматривать независимо от других.

2. Выбор количественного критерия, позволяющего выявить наилучший вариант, называемого характеристическим критерием. Критерии могут быть, в зависимости от конкретной задачи, экономического или технологического характера (минимальная стоимость, максимальная прибыль и т.д.). Независимо от того, какой критерий принят в качестве характеристического, он должен принимать максимальное (или минимальное) значение для наилучшего варианта.

3. Определение внутрисистемных переменных, через которые выражается характеристический критерий. Нужно выбрать только те

переменные, которые оказывают наибольшее влияние на характеристический критерий.

4. Построение модели, которая описывает взаимосвязь внутрисистемных переменных. Модель системы описывает взаимосвязь между переменными и отражает степень влияния этих переменных на характеристический критерий. Модель включает в себя основные уравнения материальных, производственных и других балансов; уравнения, описывающие экономические процессы в системе; неравенства, определяющие области допустимых значений переменных» [9].

«Математическая теория оптимизации включает в себя фундаментальные результаты и численные методы, позволяющие находить наилучший вариант из множества возможных альтернатив без их полного перебора и сравнения»[25].

1.2 Математические методы оптимизации

Существуют два основных класса методов оптимизации.

1. Методы безусловной оптимизации, которые позволяют найти решение на всем множестве действительных чисел.

2. Методы условной оптимизации, где на область допустимых решений налагаются определенные ограничения, формируя так называемую область допустимых решений.

В зависимости от размерности допустимого множества, методы безусловной оптимизации могут быть:

- методы одномерной оптимизации;
- методы многомерной оптимизации.

В задачах безусловной оптимизации используются численные методы поиска экстремума и методы поиска с использованием производных. Они подразделяются на:

– прямые методы, требующие только вычислений целевой функции в точках приближения;
– методы первого порядка, в которых используются вычисления первых частных производных функции;
– методы второго порядка, в которых используются вычисления вторых частных производных.

«Если ограничения на переменные не накладываются, а целевая функция является непрерывной дифференцируемой функцией, то задача называется классической задачей на экстремум (безусловный экстремум). В основе ее решения лежит теория дифференциального исчисления, в частности, теория экстремумов и опирающиеся на нее методы.»[24].

Для решения задач оптимизации с ограничениями и требованиями неотрицательности переменных применяются методы математического программирования.

Методы математического программирования классифицируются в зависимости от вида целевой функции и свойств допустимой области ограничений.

К ним относятся методы:

– линейного программирования;
– целочисленного программирования;
– нелинейного программирования;
– квадратичного программирования;
– дробно-линейного программирования;
– динамического программирования;
– параметрического программирования;
– стохастического программирования и другие.

«Методы линейного программирования используются для решения оптимизационных задач, в которых целевая функция и ограничения являются линейными функциями.» [24].

«Для решения задачи линейного программирования (далее – ЗЛП) разрабатываются специфические алгоритмы, основанные на комбинаторике, графах и т.д. Существуют эффективные методы решения ЗЛП: геометрический метод, симплекс-метод и его модификации и другие. Модификации симплекс-метода позволяют существенно сократить время счета, сделать алгоритм нечувствительным к вырожденности опорных планов, повысить размерность решаемых задач, решать так называемые блочные задачи и т.д. Кроме того, существуют способы и приемы, позволяющие расширить область применения методов решения ЗЛП. Например, некоторые нелинейные задачи могут быть преобразованы и решены методами, используемыми для ЗЛП. »[25].

В задачах целочисленного линейного программирования важную роль играют ограничения на переменные, которые требуют, чтобы значения переменных были целыми числами. Это связано с тем, что многие реальные объекты являются физически неделимыми, и простое округление до целых чисел может привести к неоптимальным решениям. Для решения таких задач используются специальные алгоритмы, включая алгоритмы Гомори, основанные на идее отсечения. Эти методы позволяют решать задачи целочисленного линейного программирования с учетом требования целочисленности переменных, что особенно важно в контексте реальных практических задач, где физические ограничения требуют точных целочисленных значений переменных.

«Интерес к целочисленному программированию обусловлен еще и тем, что многие нелинейные невыпуклые задачи могут быть сведены к ЗЛП с дополнительным требованием целочисленности. Теория целочисленного программирования используется также для разработки методов решения комбинаторных задач, например, для составления расписаний.

Частным случаем задачи целочисленного программирования являются задачи булевского программирования, где переменные принимают два значения – 0 и 1. Наиболее извест- 13 ные из этих задач – это задача о

назначениях (какого работника на какую работу поставить), задача выбора маршрута (задача коммивояжера, задача почтальона) и т.д. »[25].

«Если в задаче оптимизации целевая функция и ограничения – непрерывно дифференцируемые нелинейные скалярные функции, то имеет место задача нелинейного программирования. Универсального и эффективного метода решения задач нелинейного программирования не существует. Их класс довольно обширен. В нем выделяют отдельные категории задач, имеющих ту или иную специфику, позволяющую строить более эффективные, чем в общем случае, алгоритмы. В качестве примеров задачи нелинейного программирования можно привести квадратичные, дробно-линейные задачи и др. »[25].

В задаче квадратичного программирования целевая функция представляет собой полином второй степени, а система ограничений линейна.

В задаче дробно-линейного программирования целевая функция представляет собой дробно-линейную функцию, а система ограничений линейна.

В некоторых случаях исходные параметры задачи могут изменяться в некоторых пределах, для их решения применяется параметрическое программирование.

Среди задач оптимизации встречаются задачи, которые можно описать с помощью дискретного программирования. Дискретное программирование – это один из разделов математического программирования. Задачи дискретного программирования решаются на ограниченном множестве переменных, на которые могут принимать только дискретные значения. При условии целочисленности переменных задача дискретного программирования переходит в класс целочисленных задач.

Существуют геометрические алгоритмы решения задач оптимизации. Геометрические алгоритмы способны решать задачи не более третей размерности. Обычно – это задачи практического содержания. Примером

подобных задач чвляются задачи об оптимальном раскрое материала. Геометрические алгоритмы предусматривают перебор возможных значений искомых переменных в направлении увеличения (уменьшения) функции цели.

«В задаче стохастического линейного программирования коэффициенты целевой функции, коэффициенты в матрице коэффициентов, коэффициенты ограничений – являются случайными величинами. В этом случае сама целевая функция становится случайной величиной, и ограничения типа неравенств могут выполняться лишь с некоторой вероятностью. Приходится менять постановку самих задач с учетом этих эффектов и разрабатывать совершенно новые методы их решения» [25].

1.3 Прикладные возможности задачи оптимизации

Задачи оптимизации включают в себя широкий спектр проблем, для которых требуется найти наилучшее решение из множества допустимых вариантов. Рассмотрим несколько примеров типов задач оптимизации

Линейное программирование. Задача оптимизации, которая связана с поиском наилучшего решения линейной функции при условиях, представленных линейными неравенствами и уравнениями.

Нелинейное программирование. Включает в себя минимизацию или максимизацию нелинейной функции, часто с использованием итерационных методов для нахождения локального или глобального минимума или максимума.

Дискретная оптимизация. Задачи, связанные с поиском наилучшего решения на дискретном множестве вариантов, например, задачи коммивояжера, задачи раскроя и др.

Оптимизация с ограничениями. Включает поиск наилучшего решения с учетом различных ограничений, таких как финансовые, технические, регуляторные или другие ограничения.

Оптимизация параметров систем. Задачи, связанные с поиском оптимальных параметров системы, например, оптимизация параметров производства, управления запасами или расписания производства.

Многокритериальная оптимизация. Задачи, включающие в себя несколько критериев оптимизации, например, совместная оптимизация стоимости и времени выполнения задачи.

Задачи оптимизации, связанные с линейным программированием, являются важным классом задач, когда требуется найти максимум или минимум линейной функции при наличии линейных ограничений. Такие задачи имеют важное практическое применение и решаются с использованием методов линейного программирования.

Рассмотрим примеры практического применения линейного программирования.

Оптимизация производства. Распределение ресурсов для максимизации прибыли или минимизации затрат при ограничениях на производство.

Оптимальное планирование расписания. Например, расписание использования машин, транспортных средств, времени сотрудников с целью минимизации времени и затрат.

Оптимизация ресурсов. Например, оптимизация распределения запасов, складских ресурсов, транспорта с учетом ограничений.

Решение задачи линейного программирования требует использования специализированных методов оптимизации и математических алгоритмов. Применение этих методов позволяет найти необходимое оптимальное решение задачи линейного программирования при соответствующих ограничениях. Наиболее распростроненный метод решения линейных задач оптимизации симплексный метод

Задачи линейного программирования имеют широкие прикладные возможности и являются важным инструментом в решении сложных коммерческих и производственных задач.

Нелинейное программирование – это раздел математического программирования, который включает в себя задачи минимизации или максимизации нелинейных функций, т.е. функций, не являющихся линейными. Прикладные возможности нелинейного программирования включают широкий спектр задач различной сложности.

Оптимизация производственных процессов, включая планирование производства, управление запасами, проектирование систем снабжения и дистрибуции, оптимизация конструкций и многое другое.

В финансах и бухгалтерии может использоваться в процессах оптимизации портфеля инвестиций, финансового планирования, управления рисками, маркетинговых стратегий и прогнозирования.

В логистики может использоваться в оптимизация маршрутов доставки, планирование магистралей и транспортных сетей, управление логистическими цепочками, распределение ресурсов и т.д.

В науке и технике может использоваться в проектирование систем, оптимизация параметров приборов, прогнозирование климатических изменений, химических процессов и разработка лекарственных препаратов.

В экономики может быть использован в моделирование экономических процессов, оптимизация налогообложения, анализ рыночных тенденций, прогнозирование спроса и предложения.

Методы итерационной оптимизации широко используется для поиска оптимальных решений в различных областях, таких как инженерное дело, экономика, физика, машинное обучение и другие.

В машиностроение метод используется для оптимизации дизайна и конструкции машин и оборудования, чтобы минимизировать вес, издержки производства или максимизировать эффективность.

В области электроники методы оптимизации применяют для разработки эффективных электрических цепей, оптимизации параметров устройств или для улучшения рабочих параметров систем управления.

В финансовой сфере метод итерационной оптимизации используется для разработки торговых стратегий, оптимизации портфеля инвестиций, управления рисками и прогнозирования финансовых параметров.

В методах машинного обучения он используется для обучения моделей, поиска оптимальных параметров моделей и функций стоимости, и минимизации потерь для повышения точности и эффективности прогнозирования.

Метод итерационной оптимизации применяется для планирования маршрутов, оптимизации логистических сетей и минимизации затрат на транспортировку.

В области инженерных наук он используется для оптимизации параметров процессов, конструкций и систем, а также для решения задачи управления и спланирования.

Итерационные методы являются эффективными для поиска локальных минимумов или максимумов в задачах нелинейного программирования. Однако для гарантированного нахождения глобального экстремума часто требуется применение более сложных алгоритмов, которые могут обеспечить более широкий охват поиска.

Дискретная оптимизация является разделом оптимизации, в котором рассматриваются задачи, связанные с поиском наилучшего решения на дискретном множестве вариантов. Основным отличием дискретной оптимизации от непрерывной является то, что в дискретной оптимизации решениями могут быть только дискретные (целочисленные) значения, а не непрерывные.

Задача коммивояжера является классическим примером задачи дискретной оптимизации. Она заключается в поиске самого короткого маршрута, который проходит через все города из некоторого списка, и возвращается в исходный город. Решение этой задачи представляет собой нахождение комбинации всех возможных порядков, в которых можно посетить указанные города.

Задача о рюкзаке также является примером дискретной оптимизации. Предположим, у вас есть рюкзак определенной грузоподъемности, а также набор предметов с заданными весами и ценностями, и задача заключается в том, чтобы набрать набор предметов с максимальной ценностью, не превышая грузоподъемность рюкзака.

В задаче раскроя необходимо разместить прямоугольные или квадратные объекты на плоскости таким образом, чтобы минимизировать расход материала, например, длины сверху, ширины слева и т.д.

Графовая оптимизация решает множество задач, связанных с оптимизацией процессов на графе, таких как нахождение наибольшей клики (полного подграфа), поиск минимального остовного дерева и др.

Дискретная оптимизация играет важную роль в различных областях, таких как логистика, телекоммуникации, компьютерные науки, экономика и т.д. Она обладает широким спектром применений и является фундаментальной для многих практических проблем, где нужно принимать решения с дискретными переменными.

Оптимизация с ограничениями представляет собой задачи поиска наилучшего решения при наличии различных условий или ограничений. Эти ограничения могут быть связаны с финансовыми, техническими, регуляторными, экологическими, логистическими или другими факторами. Взаимодействие между целевой функцией и ограничениями создает сложные задачи оптимизации, которые требуют учета этих ограничений при поиске наилучшего решения.

Примерами задач оптимизации с ограничениями могут быть:

- оптимизация производственных процессов с ограничениями поставок материалов и ресурсов;
- оптимизация рабочего графика с учетом ограничений по доступности персонала и лимитов по рабочему времени;
- оптимизация инвестиционного портфеля с учетом рисков и ограничений по доступным средствам;

- оптимизация транспортной логистики с учетом ограничений по доступности транспорта и ограничений по времени доставки;
- оптимизация дизайна инженерных систем с учетом технических ограничений по материалам или спецификациям.

Решение задач оптимизации с ограничениями часто требует применения сложных методов, таких как линейное программирование, нелинейное программирование, динамическое программирование, эволюционные алгоритмы и другие методы оптимизации. Кроме того, для учета различных типов ограничений может потребоваться изучение и анализ специфических свойств каждой конкретной задачи.

Оптимизация с ограничениями возникает в самых разнообразных областях, и эти задачи имеют большое значение в поиске оптимальных решений в условиях реальных бизнес–процессов, производственных операций и стратегического планирования.

Оптимизация параметров системы относится к задачам, в которых требуется найти оптимальные значения параметров системы, чтобы достигнуть определенных целей и повысить эффективность ее функционирования. Это может быть применено в различных областях, таких как производство, управление запасами, транспортная логистика, финансы, энергетика и другие. Рассмотрим несколько примеров конкретных областей, где задачи оптимизации параметров системы имеют важное значение.

Оптимизация параметров производства. Задачи оптимизации параметров производства могут включать в себя поиск оптимальных значений производственной мощности, режимов работы оборудования, размещения оборудования на предприятии, оптимизацию производственных процессов, планирование производственных расписаний и многое другое.

Управление запасами. Оптимизация параметров системы в управлении запасами позволяет определить наилучшие стратегии управления запасами, оптимальные уровни запасов, частоту поставок, распределение запасов по складам, а также оптимальные параметры заказов.

Расписание производства. В задачах оптимизации расписания производства требуется определить оптимальное распределение производственных операций по времени, организацию рабочих смен, назначение задач машинам и сотрудникам, чтобы оптимизировать производственный процесс и минимизировать простои и затраты.

Во всех этих областях цель состоит в том, чтобы использовать методы оптимизации для поиска наилучших значений параметров системы, которые позволят достигнуть наилучших результатов в рамках заданных ограничений. В результате оптимизации параметров системы организация может повысить свою эффективность, снизить издержки и улучшить качество технологических и производственных процессов.

Многокритериальная оптимизация, также известная как задача многокритериальной оптимизации, представляет собой тип задачи оптимизации, включающий в себя несколько критериев или целевых функций, которые необходимо оптимизировать одновременно. В отличие от задач оптимизации с одним критерием, в многокритериальной оптимизации решение должно удовлетворять нескольким критериям одновременно, что делает эту задачу более сложной и интересной.

Примером задачи многокритериальной оптимизации является совместная оптимизация стоимости и времени выполнения задачи. В реальной жизни это может означать, что, например, при решении производственной задачи или разработке продукта необходимо найти такое решение, которое обеспечивает не только минимизацию стоимости, но и одновременно минимизацию времени выполнения задачи. Такое решение может быть определено как "парето–оптимальное", то есть решение, которое нельзя улучшить по одному критерию без ухудшения по другому.

Многокритериальная оптимизация имеет широкии применения в различных областях, таких как инженерия, финансы, анализ данных и исследования операций, где часто возникает потребность в нахождении

компромиссного решения, учитывающего несколько противоречивых целей сразу.

Одним из методов решения задач многокритериальной оптимизации является метод Парето–оптимальности, который позволяет находить решения, не допускающие улучшения по одному критерию без ухудшения по другому.

1.4 Применение симплексного метода в задачах оптимизации

Линейное программирование это математический метод решения задачи оптимизации, в которой требуется максимизировать или минимизировать линейную функцию от нескольких переменных, подвергаясь набору линейных ограничений.

«Линейное программирование позволяет найти экстремальные (наибольшие и наименьшие) значения линейной функции, на неизвестные которой наложены линейные ограничения. Эта линейная функция называется целевой, а ограничения, которые математически записываются в виде уравнений или неравенств, называются системой ограничений.» [21].

Решение задачи линейного программирования заключается в выборе наилучшего (оптимального) варианта из множества возможных по какому-либо признаку – критерию оптимальности.

Оптимизационная модель обычно имеет определенные признаки, которые включают в себя следующее:

Наличие целевой функции, которая является специальным показателем оптимальности или критерием оптимальности, и направлена на достижение определенных целей. Такие критерии могут быть разнообразными, включая максимизацию дохода, прибыли, производительности, эффективности или минимизацию издержек, себестоимости, капиталовложений, трудозатрат и других параметров в зависимости от конкретной задачи.

Наличие системы ограничений, которые определяют допустимое множество решений, учитывая имеющиеся ограничения, такие как наличие ресурсов, технические ограничения, бюджетные ограничения и другие факторы. В результате каждое рассматриваемое решение должно быть допустимым, то есть соответствовать имеющимся ограничениям.

Симплекс-метод - это один из основных алгоритмов линейного программирования для решения задач нахождения оптимального решения в задачах линейного программирования. Метод был разработан Свиридовым Данцигом в 1947 году. Его труды в области оптимизации и линейного программирования оказали значительное влияние на развитие прикладной математики и операционного исследования. Симплекс-метод стал мощным инструментом для решения множества задач оптимизации в различных областях, таких как производство, логистика, финансы и другие.

Симплекс-метод используется в различных областях, где требуется решение задач оптимизации и линейного программирования. Некоторые из областей применения метода включают в себя:

- производственное и операционное управление: оптимизация производственных процессов, планирование производства и управление запасами;
- логистика и транспортное планирование: оптимизация маршрутов доставки, планирование распределения ресурсов и управление логистическими сетями;
- финансы и управление рисками: оптимизация инвестиционных портфелей, управление финансовыми рисками и управление финансовыми ресурсам;.
- телекоммуникации и сетевое планирование: оптимизация распределения ресурсов в сетях связи, планирование маршрутов и управление сетевой инфраструктурой;

- маркетинг и продажи, оптимизация маркетинговых стратегий, управление ценами и промо-акциями, планирование рекламных кампаний.

Это лишь небольшой список областей, в которых симплекс-метод находит применение. В целом, метод широко используется в различных отраслях экономики, науки и бизнеса для решения сложных задач оптимизации и принятия решений.

Симплекс-метод широко используется в оптимизации производственных процессов, планировании производства и управлении запасами. В производственной сфере, где основной целью является максимизация производительности и прибыли, симплекс-метод помогает найти оптимальное распределение ресурсов и оптимальные производственные планы.

К примеру, симплекс-метод может применяться для оптимизации производственных процессов путем нахождения оптимального распределения сырья, оборудования и рабочей силы с учетом различных ограничений, таких как бюджетные ограничения, временные рамки и объемы производства.

В планировании производства симплекс-метод может помочь оптимизировать производственные циклы, определить оптимальные временные интервалы для выполнения определенных задач, минимизировать затраты на производство и максимизировать эффективность производственных процессов.

Управление запасами также может быть оптимизировано с помощью симплекс-метода. Метод позволяет оптимизировать уровень запасов, минимизировать издержки на хранение и управлять потоком материалов таким образом, чтобы удовлетворять спрос и избежать избыточных запасов.

Таким образом, симплекс-метод играет важную роль в оптимизации производственных процессов, планировании производства и управлении

запасами, обеспечивая более эффективное и экономически выгодное функционирование производственных предприятий и предприятий в целом.

Первый шаг начальное приближение:

- задача линейного программирования приводится к канонической форме: все ограничения переписываются в виде равенств, а целевая функция преобразуется в стандартную форму для максимизации или минимизации;
- строится начальное допустимое решение: выбираются базисные переменные, значения которых удовлетворяют ограничениям, и остальные переменные принимают значение нуль.

Второй шаг выбор вводимой переменной: выбирается переменная, которая будет вводиться в базис на текущей итерации. Эта переменная должна улучшить значение целевой функции. Выбор осуществляется на основе критерия наибольшего увеличения целевой функции.

Третий шаг выбор выводимой переменной. После выбора вводимой переменной находится переменная, которая должна быть выведена из базиса. Выбор осуществляется на основе критерия минимального отрицательного коэффициента.

Четвертый шаг пересчет базисной матрицы. После выбора вводимой и выводимой переменных происходит пересчет базисной матрицы с помощью метода Гаусса. Находится новое оптимальное решение задачи линейного программирования.

Пятый шаг проверка критерия оптимальности. Проверяется, достигнут ли оптимум целевой функции. Если нет, процесс повторяется с выбором новой вводимой переменной. Если оптимум достигнут и все ограничения выполнены, алгоритм завершает работу.

Заключительный шаг выход из процесса. При достижении оптимального решения и выполнении всех ограничений симплекс-метод завершает свою работу, и оптимальное решение задачи линейного программирования найдено.

Идея симплекс-метода заключается в том, что он основан на поиске оптимального решения задачи линейного программирования путем последовательного движения от одной вершины до другой по многомерному пространству ограничений. Алгоритм применяется к многограннику, образованному ограничениями задачи и ограниченному в пространстве переменных.

Идея симплекс-метода включает в себя следующие ключевые элементы.

Базисные переменные: задача линейного программирования переписывается в форму с базисными переменными, которые определяются как ненулевые переменные в допустимом решении. Остальные переменные называются не базисными и их значение равно нулю.

Переход между вершинами: симплекс-метод перемещается от одной вершины полиэдра к другой, изменяя значения базисных переменных таким образом, чтобы оптимизировать целевую функцию.

Определение оптимальности: на каждом шаге выбираются вводимые и выводимые переменные для перехода к новой вершине. Алгоритм продолжается до тех пор, пока не будет достигнуто оптимальное решение задачи.

Другими словами, симплекс-метод заключается в поиске оптимального решения, путем итеративного обхода всех вершин многогранника допустимых решений для нахождения точки, в которой целевая функция принимает максимальное или минимальное значение. Начиная с заданной вершины, метод движется вдоль ребра многогранника, по которому значение целевой функции увеличивается. Этот процесс продолжается до тех пор, пока не будет достигнута вершина, где движение вдоль любого ребра приводит к уменьшению значения целевой функции. В этой точке значение целевой функции достигает максимума или минимума, и координаты этой вершины принимаются в качестве оптимального решения линейной задачи.

1.5 Общее решение задач оптимизации в Excel

Одним из вариантом решения выбранной задачи оптимизации это использование симплекс метода в надстройке «Поиск решений» в Excel. Прежде чем перейти к самому решению разберем историю появления Excel.

«Первая в мире программа электронных таблиц— VisiCalc — создана Дэном Бриклином и Бобом Фрэнкстоном (Bob Frarikston) в 1978 году, когда в офисах еще даже не слыхивали о персональных компьютерах. VisiCalc была написана для компьютера Apple II — интересного маленького компьютера, игрушки по нынешним меркам. VisiCalc в целом стала основой будущих электронных таблиц, а также синтаксис формул до сих нор можно видеть в современных электронных таблицах. VisiCalc быстро стала востребованной, и многие дальновидные компании приобретали Apple II лишь для того, чтобы создавать свои бюджетные планы с помощью этой программы. Со временем программе VisiCalc часто ставили в заслугу, что именно она обеспечила компьютерам Apple II большую часть их первоначального успеха» [27].

«Тем временем появился новый вид персональных компьютеров; на этих ПК работала операционная система CP/M. Компания Sorcitn разработала SuperCalc— программу электронных таблиц, которая также привлекла многих последователей.

И когда в 1981 году на сцене появился компьютер IBM PC, который узаконил персональные компьютеры, то компания VisiCorp не стала медлить с переносом VisiCalc в эту новую аппаратную среду. Вскоре за ней последовала и Sorcin с версией SuperCalc, специально созданной для PC.» [27].

По современным стандартам программы VisiCalc и SuperCalc могут быть охарактеризованы как программы с ограниченными возможностями, так как они имели ряд технических ограничений. Например, ввод текста в ячейку ограничивался размерами этой ячейки, что значительно ограничивало пользователей в создании длинных заголовков или описаний. Однако,

несмотря на это, эти программы были ценным инструментом для автоматизации бюджетной работы, что позволило тысячам бухгалтеров отказаться от ручного ведения бумажных кассовых книг и освоить более гибкие компьютерные системы для обработки данных.

«Оценив успех VisiCalc, небольшая группа компьютерных гениев из компании, только что основанной в Кембридже, штат Массачусетс, усовершенствовала концепцию электронных таблиц. Руководимая Митчем Кэйпором и Джонатаном Саксом , эта организация разработала новый продукт и провела первую в компьютерной отрасли законченную маркетинговую подготовку. Я помню, как рассматривал в The Wall Street Journal рекламу Lotus 1-2-3, напечатанную большим шрифтом. Это был на моей памяти первый случай, когда в издании для широкой публики появилась реклама программного продукта. Выпущенный в январе 1983 года компанией Lotus Developmen Corporation, этот продукт имел мгновенный успех. Несмотря на этикетку с ценой 495 долларов, он по продаваемости быстро обогнал VisiCalc, взлетел на вершину продаж и оставался там многие годы.» [27].

«Электронная таблица Lotus 1-2-3 не только превосходила VisiCalc и SuperCalc всеми основными функциями, но также была первой программой, использовавшей новые уникальные возможности мощной 16-разрядной архитектуры IBM PC AT. Например, Lotus 1-2-3 игнорировала медленные вызовы DOS и передавала данные непосредственно в видеопамять, производя впечатление невероятной производительности системы, довольно необычной на то время. Прорывом была интерактивная справочная система, а хитроумные "движущиеся" панели с меню стали стандартом на многие годы. Впрочем, существовала главная возможность, которая действительно выделяла Lotus 1-2-3 среди остальных процессоров электронных таблиц. Речь идет о средстве создания макросов — поистине мощном инструменте, который предоставлял возможность пользователям электронных таблиц записывать осуществляемые ими операции и таким образом

автоматизировать многие процессы. Когда указанный макрос выполнялся, то записанные в нем операции передавались в приложение. И хотя до нынешних инструментов записи макросов было еще далеко, макросы Lotus 1-2-3 определенно были шагом в правильном направлении.» [27].

«Lotus 1-2-3 — это не только первый интегрированный пакет, но и первый успешный среди них. В нем система мощных электронных таблиц (1) сочеталась с элементарной графикой (2) и ограниченными (3), но невероятно удобными средствами управления базами данных.» [27].

«Компания Lotus постаралась, чтобы вслед за первым выпуском пакета Lotus 1-2-3 в апреле 1983 года последовал выпуск 1A. Этот новый программный продукт имел огромный успех и предоставил Lotus завидное положение единоличного монополиста на рынке процессоров электронных таблиц.» [27].

«В сентябре 1985 гола выпуск 1A был заменен выпуском 2, а в июле следующего года— выпуском 2.01, содержащим исправления выявленных ошибок. Выпуск 2, в отличие от предыдущих, имел надстройки — специальные программы, которые можно интегрировать в приложение, чтобы расширить его возможности. Кроме того, в выпуске 2 содержалась усовершенствованная система управления памятью, имелось больше функций, максимальное количество строк увеличилось в четыре раза по сравнению с предыдущими версиями. В данной версии также поддерживался математический сопроцессор и содержался усовершенствованный макроязык, популярность которого превысила самые смелые мечты его разработчиков. » [27].

«Не удивительно, что успех Lotus 1-2-3 способствовал появлению клонов — похожих в работе продуктов, в которых обычно предлагалось несколько дополнительных возможностей и которые, как правило, продавались намного дешевле. Среди более-менее заметных стоит упомянуть Twin компании Mosaic Software и серия VP Plannet компании Paperback Software. В конце концов, за нарушение авторских прав (копирование

"внешнего вида" Lotus 1-2-3) Lotus возбудила против Paperback Software судебное дело. Исход этого дела, успешный для Lotus, по существу привел к банкротству Paperback. » [27].

«Летом 1989 года Lotus выпустила DOS и OS/2-варианты долгожданной версии 3 Lotus 1-2-3. К электронным таблицам, состоящим из уже ставших привычными строк и столбцов, этот продукт добавил новое измерение; такое "расширение парадигмы" было достигнуто путем увеличения количества страниц в электронных таблицах. Впрочем, новой данная мысль в действительности не была. Идея трехмерных электронных таблиц впервые применялась в относительно малоизвестном продукте Boeing Calc, ее реализовали также в таких продуктах, как SuperCalc 5 и CubeCalc.

В версии 3 пакета Lotus 1-2-3 содержались многие полезные пользователям инструменты, которые, в конце концов, стали стандартными. Речь идет о многоуровневых электронных таблицах, одновременной работе с большим количеством файлов, их связывании, усовершенствованной графике и прямом доступе к внешним файлам баз данных. Однако в этой версии отсутствовала важная возможность, о которой мечтали многие пользователи: не была реализована высококачественный вывод.» [27].

«Версия 3 начала свою жизнь с малого рыночного потенциала, поскольку требовала для нормальной работы компьютер на базе процессора 80286 с минимальной оперативной памятью 1 Мбайт — требования довольно "жесткие" для 1989 года. И тут Lotus вытащила туз, припрятанный в ее корпоративном рукаве. Одновременно с объявлением о появлении версии 3 компания удивила буквально всех, заявив об усовершенствовании версии 2.01 (усовершенствованный продукт материализовался через несколько месяцев в виде Lotus 1-2-3 версии 2.2). Вопреки ожиданию большинства аналитиков, версия 3 не заменила версию 2. Вместо этого компания Lotus сделала блестящий ход, разбив рынок процессоров электронных таблиц на два сегмента: тот, который работает на высокопроизводительном

оборудовании, и тот, для которого по карману более скромный компьютер.» [27].

«Конечно, для фанатов электронных таблиц версия 2.2 продукта Lotus 1-2-3 панацеей не стала, но все-таки значительно расширила возможности пользователей. Самой важной из возможностей этой версии была надстройка All ways, которая предоставляла возможность "творить" привлекательные отчеты, выполненные с использованием разнообразных шрифтов, обрамлений и затенений. Кроме того, просмотр полученных результатов на экране выполнялся в режиме WYSIWYG (What You See Is What You Get — что видишь, то и получаешь). Впрочем, когда пользователи просматривали и редактировали свою работу в этом режиме, они не могли выполнять команды управления данными электронных таблиц. Но, несмотря на такое суровое ограничение, большинство пользователей Lotus 1-2-3 было вне себя от радости, потому что, имея в арсенале эту новую возможность, они наконец-то смогли создавать документы почти типографского качества. » [27].

«В мае 1990 года Microsoft выпустила Windows 3.0. Как вы, возможно, знаете, эта программа привела к изменению принципов использования персонального компьютера. Видимо, специалисты, принимавшие в Lotus решения, не считали Windows серьезным продуктом, и компания не спешила презентовать свою первую программу, работающую с электронными таблицами в Windows. Такая программа — Lotus 1-2-3 for Windows — была выпушена только в конце 1991 года. Хуже того, этот продукт, если судить объективно, оказался неудачным. Он не смог использовать преимущества среды Windows и разочаровал многих пользователей. В результате Excel, которая уже заявила о себе как о "главном" в Windows процессоре электронных таблиц, стала единоличным лидером на рынке подобных Windows-программ (и с тех пор никогда не сдавала этой позиции). Что касается Lotus, то в июне 1993 года вышла очередная ее версия: Lotus 1-2-3 версии 4 для Windows. Она была значительно лучше своего оригинала.

Версия 5 этой программы для Windows, появилась в середине 1994 года. » [27].

«В то же время Lotus выпустила версию 4.0 этого продукта для DOS (Lotus 1-2-3 Release 4.0 for DOS). Многие аналитики (и я в том числе) ожидали появления продукта, более совместимого с Windows. Однако мы ошиблись; эта версия стала лишь более усовершенствованной по сравнению с версией 3.4. Поскольку система Windows в настоящее время распространена достаточно широко, то это, скорее всего, последняя версия Lotus 1-2-3 для DOS, которая увидела свет.

Со временем электронные таблицы стали для Lotus менее важными (ее ведущим продуктом стал Notes). В середине 1995 года компания IBM приобрела Lotus Development Corporation. Появилось еще две версии Lotus 1-2-3, но это, как говорится, был тот случай, когда "и слишком мало, и слишком поздно". Excel явно доминирует на рынке процессоров электронных таблиц, а Lotus 1-2-3 продолжает терять свои позиции. » [27].

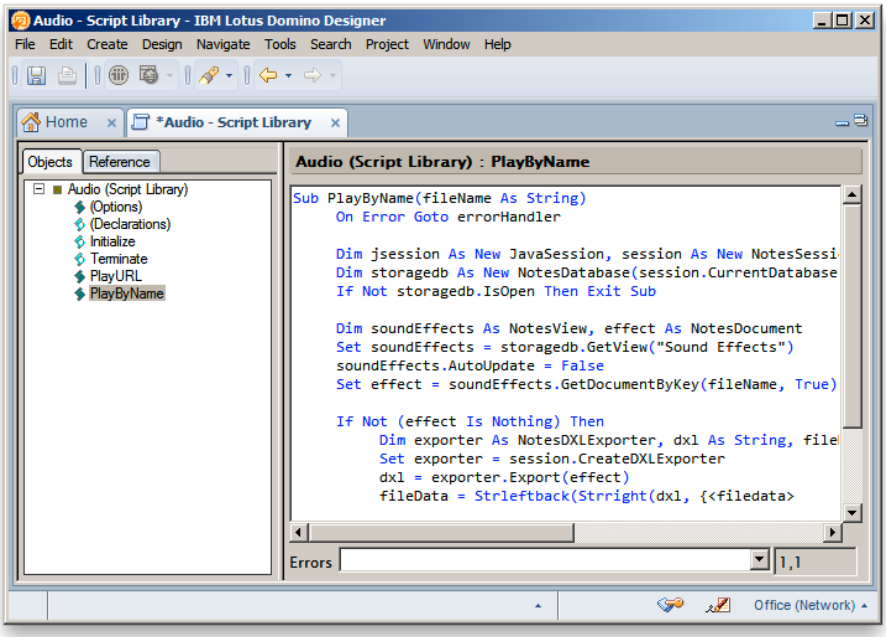


Рисунок 1 – LotusScrupt в последний версии Lotus 1-2-3

Последние версии Lotus 1-2-3 включают средство под названием LotusScript, язык сценариев, аналогичный VBA, которая представлена на рисунке 1. Однако разработчики электронных таблиц не приняли такую новость с поклонением. Если бы можно было вернуться в прошлое, то Lotus, вероятно, просто пришлось бы приобрести у Microsoft лицензию на VBA.

«В 1982 году Microsoft выпустила программу MultiPlan — свой первый продукт для работы с электронными таблицами. Предназначенная для компьютеров, которые работают под управлением операционной системы CP/M, MultiPlan вскоре была перенесен на некоторые другие платформы, в том числе, на Apple IT, Apple Ш, XENIX и MS DOS. » [27].

MultiPlan преимущественно игнорировала стандарты пользовательского интерфейса для программ. Трудная для изучения и применения, эта программа так никогда и не приобрела в США особой популярности. И не удивительно, что ее достаточно быстро обогнала Lotus 1-2-3.»

«От MultiPlan берет свое начало Excel, впервые зарекомендовавшая себя на Macintosh в 1985 году. Как и все Mac-приложения, Excel являлась графической программой (в отличие от текстовой MultiPlan). В ноябре 1987 года Microsoft выпустила первую версию Excel, предназначенную для Windows (она была названа Excel 2.0 for Windows, чтобы сохранить преемственность с номером версии, выпушенной для Macintosh). Поскольку тогда система Windows не имела широкого распространения, то в состав Excel 2.0 вошла версия Windows времени выполнения — версия, ни для чего больше не предназначенная, кроме как обеспечивать работу Excel. Менее чем через год Microsoft выпустила новую версию Excel, версию 2.1 (Excel Version 2.1). В июле 1990 года эта компания предложила небольшое обновление (2.1JC), совместимое с Windows 3.0. И хотя версии 2JC были по современным меркам довольно ограниченными и не имели привлекательного, пластичного внешнего вида последних версий, но они все равно привлекли хотя и небольшую, но верную группу поддержки и наложили прекрасный

фундамент для будущих разработок. Программа Excel имела встроенный макроязык (XLM), который состоял из функций, обрабатываемых одна за другой. Этот макроязык был достаточно мощным, но очень трудным для изучения и применения. Как вы увидите, на смену XML пришел VBA, которому и посвящена настоящая книга. » [27].

«Кроме того, Microsoft разработала версию Excel (под номером 2.20) для OS/2 Presentation Manager. Она была выпущена в сентябре 1989 года, и примерно десять месяцев спустя появилось ее обновление (версия 2.21). Впрочем, несмотря на усилия со стороны IBM, операционная система OS/2 никогда не пользовалась особой популярностью. В декабре 1990 года Microsoft выпустила Excel 3 для Windows со значительными усовершенствованиями, как внешнего вида, так и возможностей. Среди новинок были панель инструментов, средства рисования, мощный инструмент поиска решения, поддержка надстроек, поддержка связывания и внедрения объектов (Object Linking and Embedding — OLE), трехмерные диаграммы, кнопки для макросов, упрошенная консолидация файлов, редактирование в составе рабочих групп и перенос по словам текста внутри ячейки. Кроме того, в Excel 3 существовала возможность работать с внешними базами данных. Пять месяцев спустя появилось обновление Excel для OS/2. » [27].

«Версию 4, выпушенную весной 1992 года, было не только легче использовать, она также являлась более мошной и содержала больше деталей, предназначенных для опытных пользователей. «Буквально в каждом обзоре компьютерных журналов, где сравнивались процессоры электронных таблиц, Excel 4 занимала самое почетное место. Тем временем отношения между Microsoft и IBM изменились к худшему; Excel 4 для операционной системы OS/2 так никогда не была выпушена, а Microsoft прекратила выпуск версий Excel, предназначенных для этой системы. » [27].

«Версия Excel 5 предстала перед публикой в начале 1994 года и сразу заслужила восторженные отзывы. Как и ее предшественница, она попадала в

верхнюю строчку в рейтингах процессоров электронных таблиц, публиковавшихся ведущими коммерческими журналами. Несмотря на жесткую конкуренцию с Lotus 1-2-3 выпуска 5 для Windows и Quattro Pro для Windows — а ведь и тот, и другой продукт мог решить буквально каждую задачу, которую подбрасывали им электронные таблицы, — Excel 5 все равно продолжала задавать тон. » [27].

«Версия Excel 95 (известная также как Excel 7) была выпущена одновременно с Microsoft Windows 95. Microsoft специально пропустила шестой номер, чтобы у продуктов, входящих в ее пакет Office, были одинаковые номера версий. На первый взгляд. Excel 95 не во многом отличалась от Excel 5. Однако значительная часть кода ее ядра была переписана, а во многих местах наблюдалось заметное увеличение быстродействия. Важно и то, что в Excel 95 использовался тот же формат файлов, что и в Excel 5. Это был первый случай, когда усовершенствованной версии Excel не представили новый формат файла. Впрочем, до конца полной совместимость не стала, поскольку в языке VBA появилось несколько усовершенствований. Следовательно, можно было с помощью Excel 95 разрабатывать приложения, которые загружались в Excel 5 (хотя и не работали там, как положено).» [27].

«В начале 1997 года Microsoft выпустила интегрированный пакет программ Office 97, в состав которого входила Excel 97. Кроме того, Excel 97 еще называется Excel 8. Эта версия характеризовалась многими общими усовершенствованиями, а также абсолютно новым интерфейсом для разработки приложений на основе VBA. Был также предложен совершенно новый способ разработки пользовательских диалоговых окон (которые теперь назывались не диалоговыми листами, а пользовательскими формами). Microsoft попыталась сделать Excel 97 совместимым с предыдущими версиями, но эта совместимость оказалась далекой от совершенства. Чтобы многие приложения, разработанные с помощью Excel 5 или Excel 95, могли

работать в Excel 97 или более поздних версиях, приходится прибегать к определенным уловкам.

Программа Excel 2000 была выпущена в начале 1999 года; она продается как часть интегрированного офисного пакета Office 2000. Усовершенствования, которые представлены в Excel 2000, относятся, в основном, к работе в Internet, хотя несколько значительных изменений заметно и в области программирования.» [27].

«Excel 2002 появилась на рынке в середине 2001 года. «Как и предшественница, новыми возможностями, которые можно назвать серьезными, эта программа не располагает. Впрочем, появились небольшие новинки, были внесены некоторые корректировки в уже имеющиеся возможности. Вероятно, самая значительная из них — это способность восстанавливать поврежденные файлы и сохранять работу пользователя при аварийном завершении Excel. Excel будет продолжать доминировать на рынке и будет оставаться стандартом для пользователей любых уровней.» [27].

Решение задач оптимизации в Excel можно выполнить с использованием инструмента "Поиск решений". Вот пошаговое руководство для решения задачи линейной оптимизации с использованием " Поиска решений " в Excel.

1. Откройте свой файл Excel, в котором содержится задача оптимизации, и перейдите на вкладку "Данные".

2. На вкладке "Данные" выберите "Анализ" в группе "Прогнозирование" и найдите и нажмите кнопку " Поиск решений ". Если вы не видите эту кнопку, вам может потребоваться установить дополнение " Поиск решений ".

3. Когда вы нажмете на "Поиск решений", откроется окно " Параметры поиска решений ". В этом окне следует задать целевую ячейку, которую вы хотите оптимизировать, и ячейки, содержащие переменные, которые будут изменяться для достижения оптимума.

4. Выберите тип оптимизации: максимизация или минимизация. Задайте ограничения на изменение переменных: ограничения равенства, неравенства и допустимые значения.

5. Выберите метод решения задачи оптимизации. В большинстве случаев рекомендуется использовать метод "Симплекс метод".

6. Нажмите кнопку "Поиска решений" для запуска процесса оптимизации.

7. После завершения оптимизации Excel предоставит результаты, включая оптимальные значения переменных, достигнутые значение целевой функции и другие сводные данные.

Это является общим алгоритмом использования Excel для решения задач линейного программирования.

1.6 Базовая структура генетического алгоритма

Несколько ученых-программистов в середине двадцатого века независимо исследовали эволюционные системы с мыслью, что теорию эволюции представленной Чарльзом Дарвином можно использовать в качестве инструмента, позволяющего увеличить скорость и точность во многих задач разной направленности. Все они были вдохновленные естественной генетической изменчивостью и естественным отбором, и придать компьютерным алгоритмам похожие свойства, с середины 20 века и по сегодняшний день было создано множество возможных решений основанных и вдохновленной теорией эволюции Чарльза Дарвина. Давайте же рассмотрим нескольких, которые стали основой в этой области.

«Рехенберг продемонстрировал свои "эволюционные стратегии" метод в 1960-х годах, который он задействовал для оптимизации вещественных параметров для таких устройств, как аэродинамические профили. Область эволюционных стратегий является активной областью исследований по сегодняшний день, в основном развиваясь независимо от области

генетических алгоритмов (хотя в последнее время два сообщества начали взаимодействовать)» [4].

«Фогель, Оуэнс и Уолш разработали "эволюционное программирование" метод, в котором возможные решения заданных задач были представлены в виде конечных автоматов, которые были разработаны случайным образом, видоизменяя их диаграммы перехода состояний и выбирая наиболее подходящие. Несколько более широкая формулировка эволюционного программирования также остается областью активных исследований. Вместе эволюционные стратегии, эволюционное программирование и генетические алгоритмы образуют основу области эволюционных вычислений» [3].

Ко всему прочему также был ряд исследователей, которые работали в сторону разработки эволюционных алгоритмов оптимизации и машинного обучения. Эти исследователи разрабатывали, вдохновляясь тем, как работает эволюция в реальном мире. Несмотря на все приложенные усилия, им не хватило чуточки удачи, в том, чтобы их работы стали не только известными обществу, но еще и были развиты, кем-то кто бы двигался в том же направление. Следовательно их работы и не получи такого развития как работа Рэхенберга по «эволюционным стратегия, Фогеля, Оуэнса и Уолша по эволюционному программированию, а так же генетическим алгоритмам Джонна Холланда.

Так же существовало сообщество биологов-эволюционистов, которые использовали электронно-вычислительные машины в своих работах для моделирования процесса эволюции с целью взять в свои руки этапы экспериментов. Не лишним будет сказать, что в дни становления компьютеров отчётливо слышались тенденции использования эволюционных вычислений.

«Холланд вместе с коллегами и студентами разработали генетические алгоритмы (ГА) в 1960-х и 1970-х годах, когда работал преподавателем в Мичиганском университете. Первоначально Холланд поставил себе цель не в

разработке алгоритмов для решения конкретных задач, а скорее в формальном изучении феномена адаптации. Он рассматривал, как адаптации происходит в природе, и разработал способы, с помощью которых механизмы естественной адаптации могли бы быть импортированы в компьютерные системы» [1].

«Генетический алгоритм − это метод перехода от одной популяции "хромосом" (например, цепочек единиц и нулей или "битов") к новой популяции с использованием своего рода "естественного отбора" вместе с вдохновленными генетикой операторами кроссовера, мутации и инверсии. Каждая хромосома состоит из "генов" (например, битов), каждый ген является экземпляр определенного "аллеля" (например, 0 или 1). Оператор отбора выбирает те хромосомы в популяции, которым будет разрешено размножаться, и в среднем более подходящие хромосомы дают больше потомства, чем менее подходящие. Кроссинговер обменивает участки двух хромосом, грубо имитируя биологическую рекомбинацию между двумя однохромосомными ("гаплоидными") организмами; мутация случайным образом изменяет значения аллелей в некоторых местах хромосомы; а инверсия изменяет порядок смежных участков хромосомы. хромосому, таким образом перестраивая порядок, в котором расположены гены» [29].

«Внедрение Холландом популяционного алгоритма с кроссовером, инверсией и мутацией было крупным нововведением. (Эволюционные стратегии Рехенберга начинались с "популяции" из двух особей, одного родителя и одного потомства, причем потомство было мутировавшей версией родителя; многоиндивидуальные популяции и скрещивание были включены только позже. Эволюционное программирование Фогеля, Оуэнса и Уолша аналогично использовало только мутацию для обеспечения вариации)» [3]. Ко всему прочему, Холланд был первым, кто попытался поставить вычислительную эволюцию на прочную теоретическую основу, выпустив книгу «Адаптация в естественных и искусственных системах» и дав теоретическую основу для феномена адаптации в рамках генетического

алгоритма. До начала двухтысячных этот теоретический фундамент, основанный на понятии его «схемы», составляла основу почти всех последующих теоретических работ по генетическим алгоритмам.

Чтобы разобраться в работе генетического алгоритма, а также их вариация после того как немного познакомились с историей возникновения генетического алгоритма разберем основные понятия.

Вектор — может быть представлен в виде строки состоящих их упорядоченного набора чисел, поэтому понятия вектора и строки часто считаются идентичными. Числа содержащиеся в векторе называются элементами вектора.

Булев вектор — это такой вектор, в котором компоненты принимают значения из двух элементного (булева) множества, например, {0, 1} или {−1, 1}.

Хеммингово расстояние — используется для вычисления числа различных элементов в двух булевых векторов, в дальнейшем для краткости, возможно, будет записано как просто расстояние.

Хеммингово пространство — «пространство булевых векторов, с введенным на нем расстоянием (метрикой) Хемминга. В случае булевых векторов размерности n рассматриваемое пространство представляет собой множество вершин n-мерного гиперкуба с хеммингово метрикой. Расстояние между двумя вершинами определяется длиной кратчайшего соединяющего их пути, измеренной вдоль ребер» [1].

Хромосома — «вектор (или строка) из каких-либо чисел. Если этот вектор представлен бинарной строкой из нулей и единиц, например, 1010011, то он получен либо с использованием двоичного кодирования, либо кода Грея» [1].

«Ген — каждая позиция (бит) в хромосоме.

Индивидуум (особь, генетический код) — набор хромосом (вариант решения задачи)» [1]. В большинстве случаев особь состоит из одной хромосомы, поэтому будем считать особь и хромосому, идентичные понятия.

Кроссинговер (скрещивание) — функция, отвечающия за обмен двумя индивидуумами частями своего генетического кода. Пример представлен на рисунке 2.

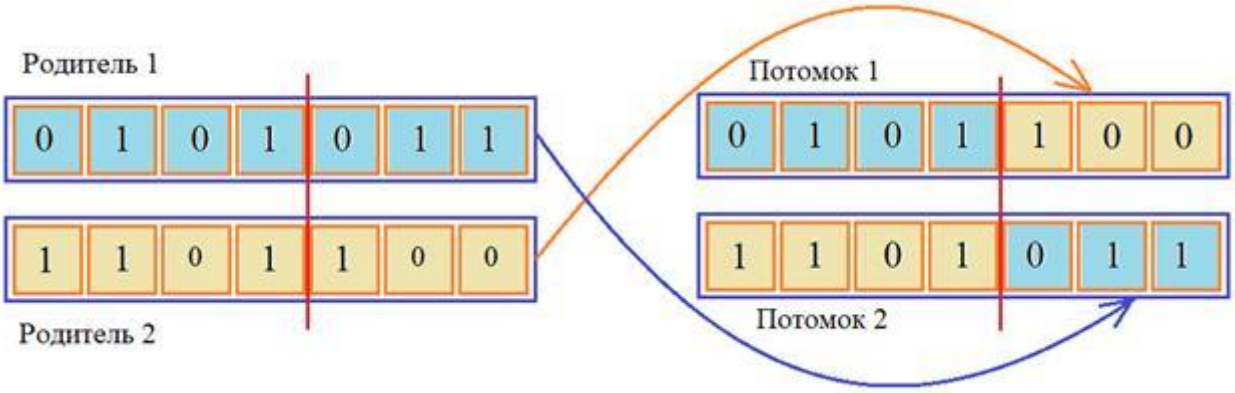


Рисунок 2 – Пример скрещивания хромосом

Для полного понимания кроссинговера рассмотрим существующие несколько видов.

Одноточечный кроссинговер это скрещивание, в котором родительские хромосомы разделяются только в одной случайной точке. Рассмотрим его работу в общем виде.

«Пусть имеются две родительские особи с хромосомами $X = \{x_i, i \in [0; L]\}$ и $Y = \{y_i, i \in [0; L]\}$. Случайным образом определяется точка внутри индивидуума (точка разрыва), в которой оба индивидуума делятся на две части и меняют их между собой местами» [5]. Общий вид представлен на рисунке 3.

Родители		Потомки
$x_1x_2x_3 \ldots x_{n-1}x_n \ldots x_m$	$\rightarrow$	$x_1x_2x_3 \ldots x_{n-1}y_n \ldots y_m$
$y_1y_2y_3 \ldots y_{n-1}y_n \ldots y_m$	$\rightarrow$	$y_1y_2y_3 \ldots y_{n-1}x_n \ldots x_m$

Рисунок 3 – Общий вид одноточечного кроссинговер

«В двухточечном кроссинговере (и многоточечном кроссинговере) хромосомы рассматриваются как циклы, которые формируются соединением концов линейной хромосомы вместе. Для замены сегмента одного цикла сегментом другого цикла требуется выбор двух точек разреза. В этом представлении, одноточечный кроссинговер может быть рассмотрен как кроссинговер с двумя точками, но с одной точкой разреза, зафиксированной в начале строки» [6]. Из этого можно сделать вывод, что двухточечный кроссинговер решает задачу более полно, в отличие от одноточечного скрещивания. Хромосома в двухточечном кроссинговере, рассматривается как цикл. Этот цикл может состоять из значительного количества сегментов, из-за того что может случиться «циклический возврат» содержащий в себе большое количество сегментов. Пример того как это происходит показан на рисунке 4.

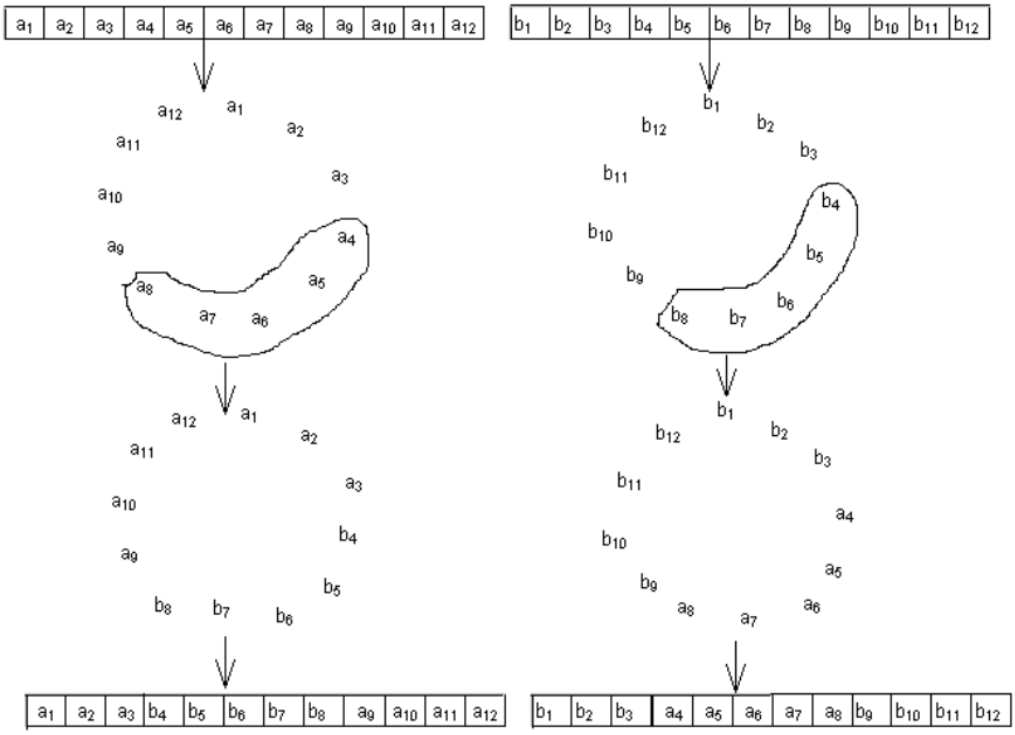


Рисунок 4 – Возврат цикла в двухточечном кроссинговере

«Для многоточечного кроссинговера выбираем m точек разреза $k_i \in \{1, 2, \ldots, N_{var}\}, i = 1 : m, N_{var}$ — количество переменных (генов) в особи. Точки разреза выбираются случайным образом без повторений и сортируются в порядке возрастания. При кроссинговере происходит обмен участками хромосом, ограниченными точками разреза и таким образом получают двух потомков. Участок особи с первым геном до первой точки разреза в обмене не участвует» [1]. Рассмотрим приме работы трёхточечного кроссинговера, который является видом многоточечного кроссинговера. Пусть имеются две особи по хромосоме, состоящей из одиннадцати двоичных генах. Пример особей рассмотрен в таблице 1.

Таблица 1 – Особи для многоточечного кроссинговера

Особь 1	1	0	1	0	1	1	0	1	1	0	0
Особь 2	1	1	0	0	0	1	1	0	0	1	1

Случайным образом выберем количество и значения точек разреза кроссинговера, представим выбор в таблице 2.

Таблица 2 – Координаты точки разреза

Точка разреза (m = 3)	3	5	10

Создадим двух новых потомков. В таблице 3, представлены 2 созданных потомка.

Таблица 3 – Хромосомы потомков

Потомок 1	1	0	1	0	0	1	0	1	1	0	1
Потомок 2	1	1	0	0	1	1	1	0	0	1	0

«Однородный кроссинговер создает маску (схему) особи, в каждом локусе которой находится потенциальная точка кроссинговера. Маска кроссинговера имеет ту же длину, что и скрещивающиеся особи. Создать маску можно следующим образом: введем некоторую величину $0 < p_0 < 1$, и если случайное число больше p_0, то на n-ю позицию маски ставим 0, иначе — 1» [1]. Получается, элементы маски представляют собой случайные набор чисел из 0 и 1. Согласно этим значениям, геном потомка становится первая (если ген маски = 0) или вторая (если ген маски = 1) особь-родитель.

В таблице 4, рассмотрим особи в качестве примера.

Таблица 4 – Рассматриваемые особи

Особь 1	1	0	1	0	1	1	0	1	1	0	0
Особь 2	1	1	0	0	0	1	1	0	0	1	1

Для каждого создаваемого потомка создадим маску из 11 случайно выбранных элементов из множества {0; 1}. В таблице 5 представлены полученные маски.

Таблица 5 – Маски

Маска 1	1	0	0	1	0	0	1	0	1	0	0
Маска 2	0	0	1	0	1	1	0	1	0	1	0

Создадим потомков по следующему правилу: «если на i-ом месте в соответствующей маске стоит 1, то ген 1 родителя переходит потомку, иначе наследуется ген второго родителя» [1]. В таблице 6 представлены полученные потомки.

Таблица 6 – Потомки

Потомок 1	1	0	1	0	1	1	1	1	0	0	0
Потомок 2	0	0	1	1	0	0	0	0	0	0	0

Однородный кроссинговер очень похож на многоточечный, но хромосома случайных битовых значений в нем длиннее. Это гарантирует, что в потомках будут чередоваться короткие строки особей-родителей. «Алгоритм однородного кроссинговера для двоичных строк полностью идентичен дискретному воспроизведению для вещественных хромосом. Такой вид кроссинговера еще называют унифицированным» [15].

Мутация — «случайное изменение одной или нескольких позиций в хромосоме, которое может быть унаследовано потомками данного индивидуума» [2].

Изменения необходимые для вывода популяции из локального экстремума во время попадания в оное и препятствует преждевременной сходимости, как раз и достигаются с помощью мутации.

«Мутация позволяет выбирать для изменения несколько ген в хромосоме, причем их число также может быть случайно выбранным. Следует мутация, как и кроссинговер, могут проводиться не только по одному случайному гену, а сразу изменяя некоторую группу подряд идущих точек. Вероятность мутации p_m (как правило, $p_m \leq 1$) может являться или фиксированным случайным числом от 0 до 1, или заданной какой-то конкретной функцией от какой-либо характеристики решаемой задачи. Например, можно положить вероятность мутирования генов, равную сумме особей в популяции, деленной на количество чисел всех генов равной единицы. Проведя множество исследований и вычислительных экспериментов, а после, сравнив результаты, оптимальное значение вероятности мутации для всех задач не было выявлено» [1]. Из этого сделали вывод, что для разных задач и функций подходят разные вариации

вероятности мутации, для задач использующих унимодальные функции отлично подойдет мутация с фиксированной вероятностью. Для мульти модальных функций, лучше использовать вероятность заданную конкретной функций [9]. Существуют разные разновидности мутаций, в работе далее рассмотрим один из существующих вариантов.

Двоичная мутация. «Для особей, кодированных двоичным кодом или кодом Грея, мутация заключается в случайном инвертировании гена (0 заменяется 1 и наоборот). Эффект мутации зависит от примененного способа кодирования генов. Так, в одних задачах при мутации наилучший эффект достигается в случае, когда особи закодированы кодом Грея, а в других — с помощью двоичного кода» [1]. На рисунке 5 представлен пример двоичной мутации.

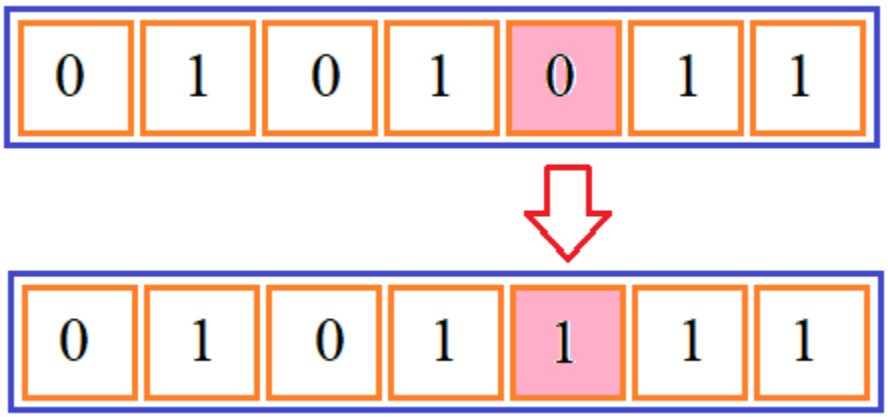


Рисунок 5 – Мутация

Инверсия — «действие изменяющая порядок следования ген в хромосоме или в ее фрагменте» [2].

Популяция — совокупность индивидуумов размера N. На рисунке 6 показан пример популяции в общем виде.

Рисунок 6 – Популюция в общем виде

Важным моментом является генерация новой популяции, опираясь на старую популяцию одним их разновидностей отбора [10]. Далее рассмотрим несколько основных методов

«При отборе усечением используют популяцию, состоящую как из особей-родителей, так и особей потомков, отсортированную по возрастанию значений функции пригодности особей. Число особей для скрещивания выбирается в соответствии с порогом $T \in [0; 1]$. Порог определяет, какая доля особей, начиная с самой первой (самой пригодной), будет принимать участие в отборе. В принципе, порог можно задать и числом больше 1, тогда он будет просто равен числу особей из текущей популяции, допущенных к отбору. Среди особей, попавших «под порог», случайным образом выбирается одна и записывается в новую популяцию. Процесс повторяется N раз, пока размер новой популяции не станет равен размеру исходной популяции. Новая популяция состоит только из особей с высокой пригодностью, причем одна и та же особь может встречаться несколько раз, а

некоторые особи, имеющие пригодность выше пороговой, могут не попасть в новую популяцию» [17].

Сортировка применяемая в этом отборе значительное увеличивает время работы алгоритма при работе с популяцией большого размера, а так же затрата времени сильно зависит от самого алгоритма сортировки в целом.

Элитарный отбор, отбор при котором создается промежуточная популяция, которая включает в себя как потомков, так и их родителей. «Члены этой популяции оцениваются, а за тем из них выбираются N самых лучших (пригодных), которые и войдут в следующее поколение. Зачастую данный метод комбинируют с другими — выбирают в новую популяцию, например, 10 % «элитных» особей, а остальные 90 % — одним из традиционных методов селекции. Иногда эти 90 % особей создают случайно, как при создании начальной популяции перед запуском работы генетического алгоритма» [2].

«В отбор вытеснением выбор особи в новую популяцию зависит не только от величины ее пригодности, но и от того, есть ли уже в формируемой популяции особь с аналогичным хромосомным набором. Отбор проводится из числа родителей и их потомков. Из всех особей с одинаковой приспособленностью предпочтение сначала отдается особям с разными генотипами. Таким образом, достигаются две цели: во-первых, не теряются лучшие найденные решения, обладающие различными хромосомными наборами, во-вторых, в популяции постоянно поддерживается генетическое разнообразие. Вытеснение в данном случае формирует новую популяцию скорее из удаленных особей, вместо особей, группирующихся около текущего найденного решения. Данный метод наиболее пригоден для многоэкстремальных задач, при этом помимо определения глобальных экстремумов появляется возможность выделить и те локальные максимумы, значения которых близки к глобальным» [1]. На рисунке 7 представлен элитарный отбор в общем виде.

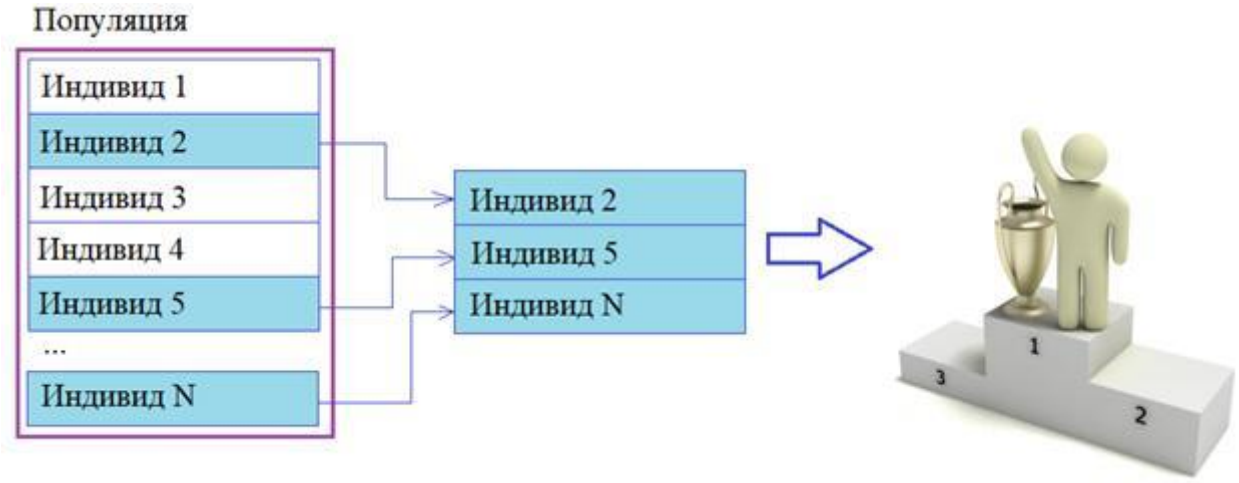


Рисунок 7 – Элитарный отбор

Пригодность (приспособленность) — «значение вычисляемая заданной ранее функции или критерии» [1].

Аллель — «совокупность подряд идущих генов» [1].

Локус — «позиция гена в хромосоме» [1].

Эпистаз — «влияние гена на пригодность индивидуума в зависимости от значения гена, присутствующего в другом месте. Ген считают эпистатическим, когда его присутствие подавляет влияние гена в другом локусе. Эпистатические гены из-за их влияния на другие гены иногда называют ингибирующими. Подавление проявления гена неаллельным ему геном называется гипостазом, а сам подавляемый ген — гипостатическим» [1].

Познакомившись с определениями, поймём, что не следует сильно путать термины генетического алгоритма с терминами биологии генетики, так как они имеют только отдаленную смысловую схожесть хоть и обозначаются одни и теме словами. Поэтому часто принято людьми работающими в сфере генетического алгоритма называть одно и тоже определения либо биологическим термином, либо математическо-информационным.

«В биологических системах полный генетический пакет называется генотипом. В искусственных системах полный генетический пакет строк называется структурой.

В биологических системах в процессе индивидуального развития организма взаимодействие генотипа с окружающей средой формирует совокупность внешних признаков и свойств, называемую фенотипом.

В математическом моделировании рассматриваемая структура декодируется с помощью множества параметров, которое в литературе иногда называют альтернативным решением или точкой. Всевозможные значения параметров образуют пространство решений. В искусственной генетической системе возможно использование как числовых, так и нечисловых параметров» [22].

Для определения гена волос используется 15 различных локусов, как в пример черный цвет имеет 15-аллельное значение, это был хороший пример специализированных генах. Таким образом, и в генетическом алгоритме вполне может быть специализированные гены у индивидуумов. «В терминологии генетического алгоритма говорится, что строки образованы значениями функции, или детекторами. Значения функции могут быть локализованы в различных позициях строки» [14].

Основной принципы работы генетического алгоритма представлен в блок-схеме на рисунке 8.

Рисунок 8 – Блок-схема генетического алгоритма

Критерием окончанием остановки генетического алгоритма может служить заданное ранее количество поколений, либо заданное ранее количество схождений популяции. «Схождение это состояние популяции, когда все строки популяции почти одинаковы и находятся в области некоторого экстремума» [1]. «В такой ситуации оператор скрещивания очень слабо или вообще никак не изменяет популяцию, так как создаваемые при нем потомки представляют собой копии родителей с перемененными участками хромосом» [2]. В следствии получившиеся за счет мутации индивидуумы склонны не сохранять потомство и вымирать, так как чаще имеют меньшую приспособленность, особенно если данный экстремум является глобальным максимумом.

«Основными функциями (операторами) генетического алгоритма являются кроссинговер, мутация, выбор родителей и селекция (отбор хромосом в новую популяцию)» [7].

Следует, в разработке собственного метода генетического алгоритма для решения конкретной задачи необходимо верно выбрать и использовать основные операторы. В данной работе в дальнейшем будет взят уже существующий метод генетического алгоритма для решения задачи OneMax и модифицирован под задачу.

Следует от того какой вид имеют функций в построенном генетическом алгоритме, зависит эффективность решения конкретных задач а также ее реализация. Таким образом, правильное использование и модернизация основных форм операторов ведет к получению генетического алгоритма, который нам понадобиться в решение нашей задачи максимизации общего годового дохода без превышения бюджета строительства.

Из этого вытекает гипотеза, использование генетического алгоритма при решении задачи выбора плана застройки участка, может ускорить получение варианта, который принесет достаточный доход.

Глава 2 Постановка исследовательской задачи и выбор программных средств для решения задачи

2.1 Постановка исследовательской задачи

Опишем работу генетического алгоритма на примере решения задачи оптимизации, сформулированной ниже.

Имеется m строительных проектов зданий и n участков под застройку (n> m). Для осуществления каждого проекта требуется выбрать один из участков Предварительные исследования определили необходимые инвестиции (затраты) и чистый годовой доход для каждой пары: строительный объект и участок под застройку.

Компания располагает определенным инвестиционным бюджетом. Требуется определить оптимальный план застройки: максимизировать доход при ограничении на бюджет.

Построим математическую модель и математическую модель задачи размерности 5X4 с данными описаными в таблице. Каждому из четырех проектируемых зданий поставим в соответствие букву алфавита: A, B, C, и D, а каждому участку присвоим номер: 1; 2; 3; 4; 5. В таблицу 7 внесем возможные затраты для каждой пары (проект здания; номер участка). В таблицу 8 внесем доходы, которые могут быть получены при выборе пары (проект здания; номер участка).

Таблица 7 – Затраты

Затраты	A	B	C	D
1	23	48	48	43
2	45	31	36	35
3	46	50	38	45
4	47	22	41	23
5	39	21	32	45

Таблица 8 – Доходы

Доходы	A	B	C	D
1	7	30	10	16
2	29	14	28	21
3	16	23	29	12
4	8	24	9	12
5	24	27	13	11

Введем обозначения.

a_{ij} (усл.д.е.) – доход, получаемый при продаже здания, построенного на i-том участке по j-том проекту

b_{ij} (усл. д.е.) – затраты на строительство здания, на i-том участке по j-тому проекту.

d (усл. д.е.) – экспертная оценка минимально возможного бюджета.

$x_{ij} \in \{0; 1\}$ $(i = \underline{1,5}\,; j = \underline{1,4})$. Если $x_{ij} = 1$, то это означает, что под $j-$ый проект выбран i -участок.

Для каждого из проектов определим, на каком из участков затраты будут минимальны $b_i = \min b_{ij}$ $(i = \underline{1,5}\,; j = \underline{1,4})$

Вычислим минимальный суммарный бюджет необходимый для строительства всех зданий при минимальных затратах на строительство каждого отдельного здания.

$b = \sum_{i=1}^{5} b_i$

Построим математическую модель задачи.

$$F(x) = \sum_{i=1}^{5} \sum_{j=1}^{4} a_{ij} x_{ij} \rightarrow max \qquad (1)$$

$$\sum_{j=1}^{4} x_{ij} = 1 \ (i = 1,5); \qquad (2)$$

$$\sum_{i=1}^{5}\sum_{j=1}^{4} b_{ij}x_{ij} \leq d; \quad (3)$$

$$\sum_{i=1}^{5}\sum_{j=1}^{4} b_{ij}x_{ij} \geq b; \quad (4)$$

$$b \leq d. \quad (5)$$

$F(x)$ в равенстве (1) – функция цели, суммарный доход в условных денежных единицах, получаемый при реализации всех четырех проектов

Ограничение (2) означает, что под заданный проект может быть выбран только один участок.

Ограничение (3) означает, что суммарные затраты на осуществление всех проектов не должны превышать значения d.

Ограничения (4), (5) означают, что суммарные затраты на осуществление всех проектов не могут быть меньше, чем минимально возможный бюджет b.

2.2 Решение тестируемой задачи в Excel

Задача может быть легко решена в EXCEL, с помошью надстройки принятия решений.

Запишим исходные данные задачи, внесем в таблицу числовые значения. В ячейках A1:E6 и G1:K6 представлены соответственно расход и доход, получаемые при продаже строений.

На рисунке 9 выделена зона, в которой находятся таблицы.

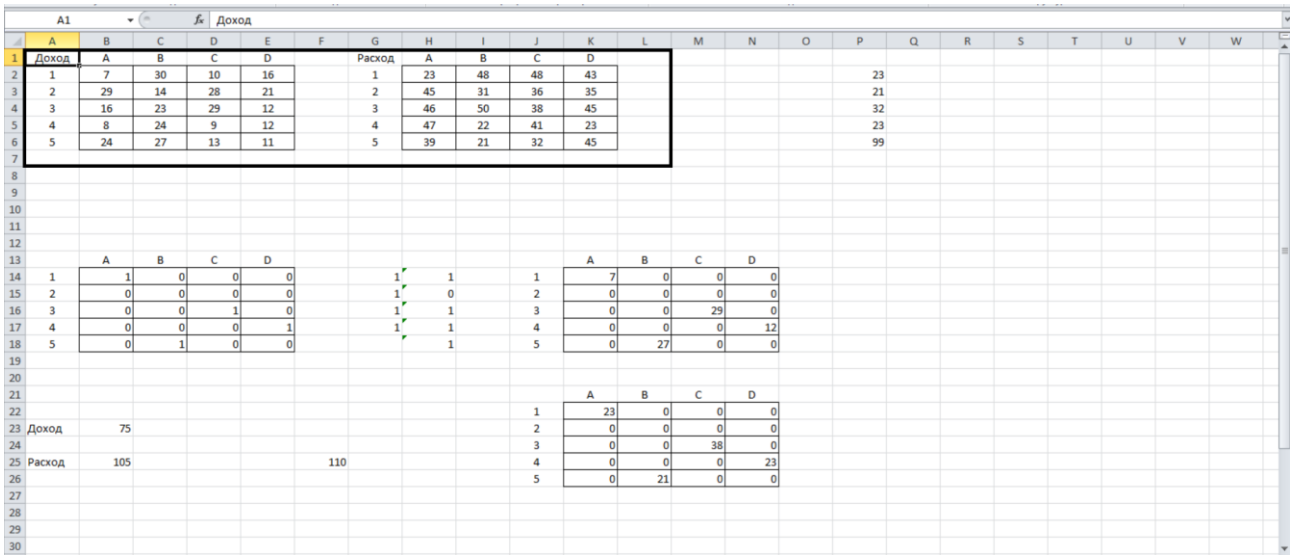


Рисунок 9 – Зона исходных данных

Вычислим минимальный бюджет, используя функцию МИН. Определяем минимальные значения в каждом столбце таблицы расходов и суммируем. Формула для суммирования по столбцу имеет вид МИН(H2:H6). Для суммирование используем СУММ(P2:P5). На рисунке 10 выделена зона расчёта минимального бюджета.

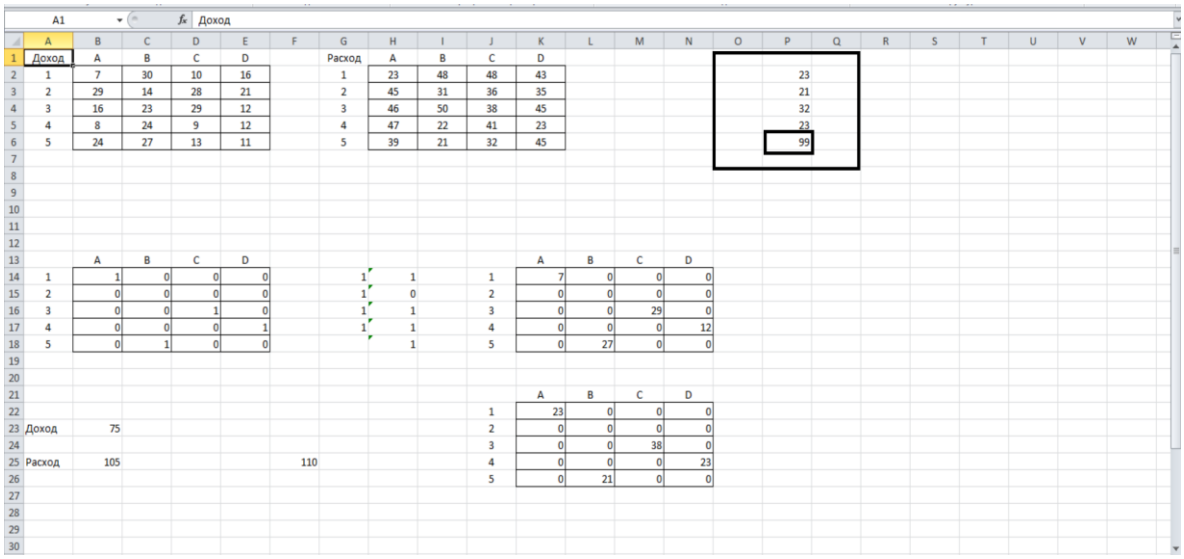


Рисунок 10 – Зона расчёт минимального бюджета

Создадим вспомогательные таблицы для вычислений. В таблицу K14: N18 будут расчеты дохода , а в таблице K22: N26 будет расчет затрат. На рисунке 11, выделена зона для вспомогательного расчёта дохода и затрат.

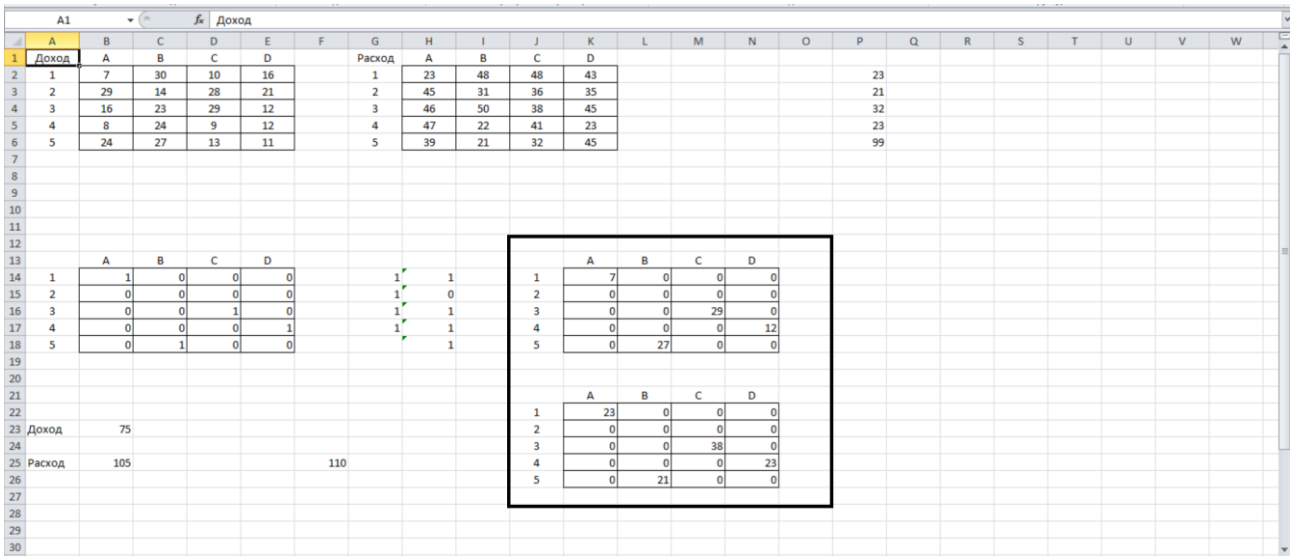


Рисунок 11 – Зона для расчета затрат и дохода

Внесем оставшиеся ограничения и воспользуемся надстройкой «Поиск решений». Решение с заданной линейной целевой функцией и заданными линейными ограничениями буде построено на основе симплекс-метода. На рисунке представлены параметры поиска решения для нашей задачи.

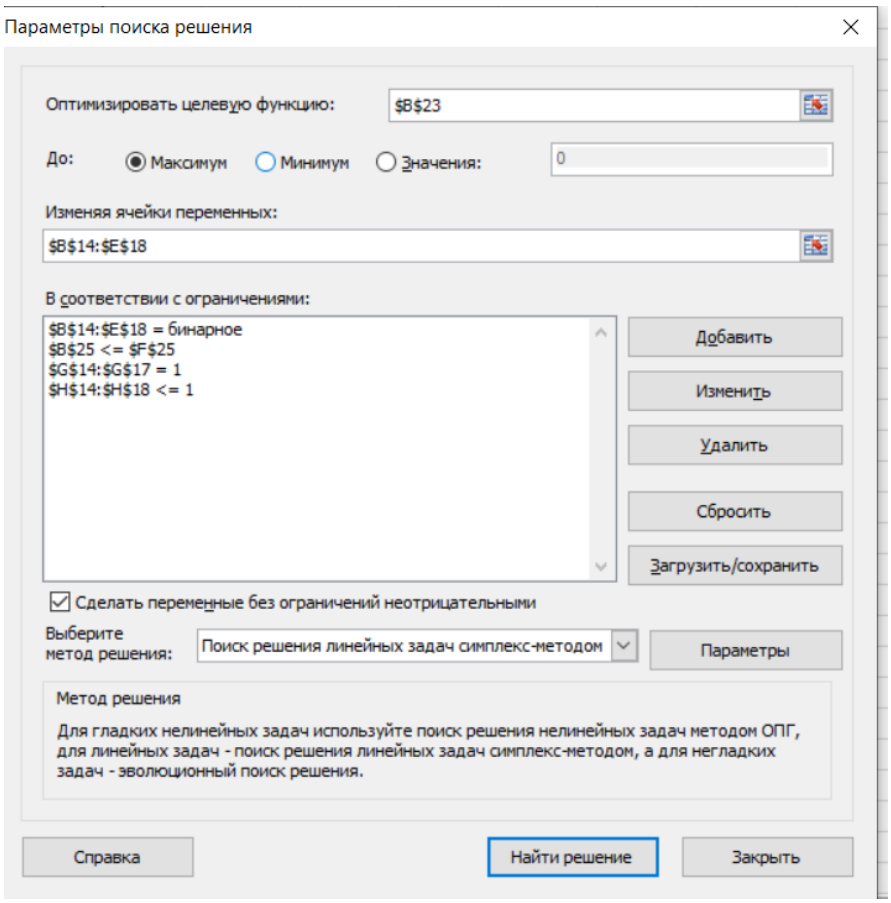
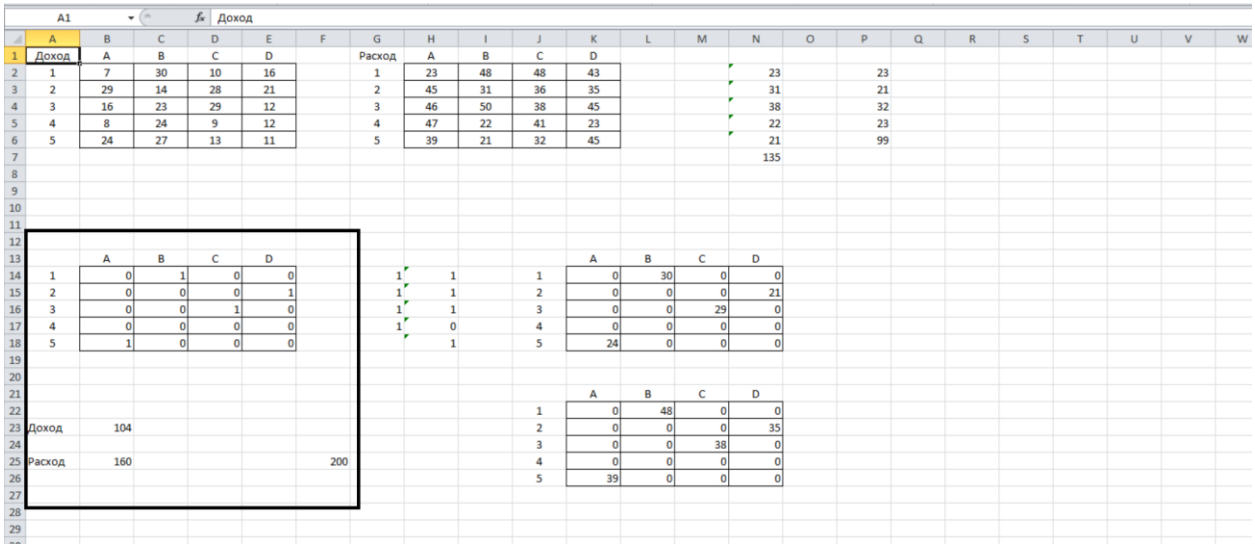


Рисунок 12 – Параметры поиска решения

На рисунке 13 выделена зона с окончательными результатами решения задачи. При решение заадчи получили доход раный 75, расход равный 160 при бюджете 200.



Доход

	A	B	C	D
1	7	30	10	16
2	29	14	28	21
3	16	23	29	12
4	8	24	9	12
5	24	27	13	11

Расход

	A	B	C	D
1	23	48	48	43
2	45	31	36	35
3	46	50	38	45
4	47	22	41	23
5	39	21	32	45

23	23		
31	21		
38	32		
22	23		
21	99		
135			

	A	B	C	D
1	0	1	0	0
2	0	0	0	1
3	0	0	1	0
4	0	0	0	0
5	1	0	0	0

1	1	
1	1	
1	1	
1	0	
	1	

	A	B	C	D
1	0	30	0	0
2	0	0	0	21
3	0	0	29	0
4	0	0	0	0
5	24	0	0	0

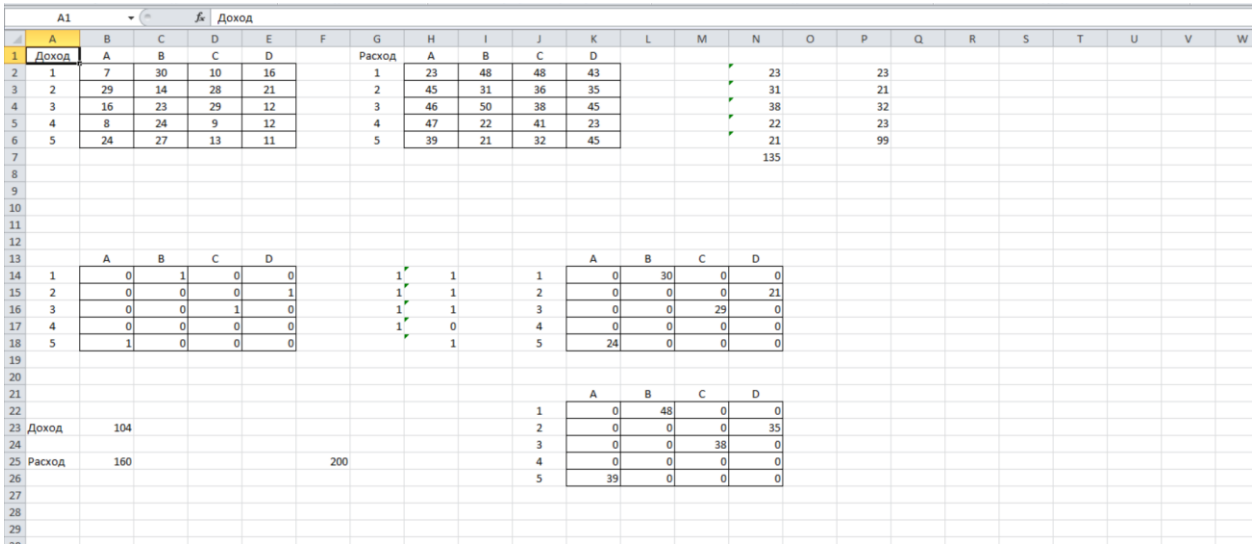
	A	B	C	D
1	0	48	0	0
2	0	0	0	35
3	0	0	38	0
4	0	0	0	0
5	39	0	0	0

Доход 104

Расход 160 200

Рисунок 13 – Зона с окончательными результатами

На рисунке 14 представлена полностью решёная задача.



Доход

	A	B	C	D
1	7	30	10	16
2	29	14	28	21
3	16	23	29	12
4	8	24	9	12
5	24	27	13	11

Расход

	A	B	C	D
1	23	48	48	43
2	45	31	36	35
3	46	50	38	45
4	47	22	41	23
5	39	21	32	45

23	23		
31	21		
38	32		
22	23		
21	99		
135			

	A	B	C	D
1	0	1	0	0
2	0	0	0	1
3	0	0	1	0
4	0	0	0	0
5	1	0	0	0

1	1	
1	1	
1	1	
1	0	
	1	

	A	B	C	D
1	0	30	0	0
2	0	0	0	21
3	0	0	29	0
4	0	0	0	0
5	24	0	0	0

	A	B	C	D
1	0	48	0	0
2	0	0	0	35
3	0	0	38	0
4	0	0	0	0
5	39	0	0	0

Доход 104

Расход 160 200

Рисунок 14 – Решение задачи в Excel

Как видим, данного вида задачи легко решается в Excel с помощью поиска решений. Но в Excel есть ограничение в размерности задачи, что не позволяет решать задачи большой размерности.

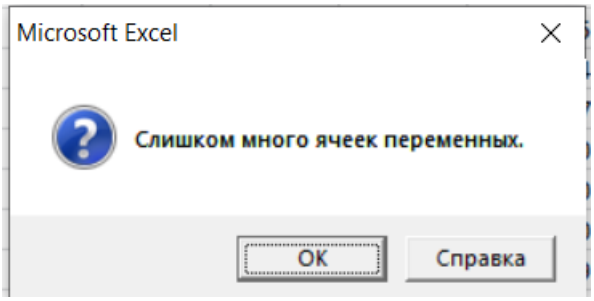


Рисунок 15 – Ошибка в Excel

На рисунке 15 представлена ошибка, которая появлется, если попытаться решить задачу размерностью более 20x10. Для задач большой размерности необходимы другие математические методы, позволяющие справиться с проблемой.

2.3 Обоснование выбора программных средств для решения поставленной задачи

В современном мире существует множество языков программирования высокого уровня, которые постоянно развиваются в связи с активным прогрессом информационных технологий и цифровой экономики. Этап этого развития также связан с появлением новых систем программирования, обусловленных такими технологиями, как искусственный интеллект, интернет вещей, большие данные и другими.

Из-за этих изменений происходит усложнение процесса выбора языка программирования, так как языки меняются, а также меняются и технологии, которые они поддерживают. Кроме того, проекты в области информационных технологий становятся все более сложными, что дополнительно усложняет задачу выбора оптимального языка программирования для их реализации. Не всегда удается правильно выбрать язык, наилучшим образом подходящий для решения определенных задач, что

может снизить эффективность реализации проектов в области информационных технологий.

Лидерами последних лет среди языков объектно-ориентированного программирования являются C++, C#, JavaScript [8], [11], [30]. Несмотря на это, существует достаточно большой перечень языков программирования высокого уровня из категории объектно-ориентированных, в том числе:

- C#;
- C++;
- Java;
- Delphi;
- Eiffel;
- Simula;
- Object Pascal;
- Visual DataFlex;
- Perl;
- PowerBuilder;
- Python;
- ActionScript (3.0);
- JavaScript;
- JScript .NET;
- Ruby;
- Smalltalk;
- PHP и другие.

Для написания программы, использующий генетический алгоритм в качестве основы, был выбран язык высокого уровня Python. Этот язык чаще всего используют для на писания программ с использованием искусственого интелекта и эволюционных алгоритмов [18], [19].

Существует множество IDE, предназначеных для написания программного кода на Python. Ниже перечислены десять современных IDE.

PyCharm. Один из самых популярных IDE для Python, предлагает широкий спектр функций и инструментов для разработки.

Visual Studio Code. Мощный и гибкий редактор кода, который поддерживает различные языки программирования, включая Python.

Spyder. Научно-ориентированная IDE с удобными инструментами для анализа данных и визуализации в Python.

Jupyter Notebook. Интерактивная среда для работы с Python, используется в основном для анализа данных, визуализации и научных вычислений.

Sublime Text. Легкий редактор кода, который обладает большим сообществом пользователей и поддерживает различные языки программирования, включая Python.

Atom. Гибкий и расширяемый текстовый редактор, разработанный командой GitHub, который хорошо поддерживает Python.

Thonny. Простая и легкая в использовании IDE, специально разработанная для обучения Python начинающих программистов.

GNU Emacs. Мощный текстовый редактор, возможности которого могут быть расширены для работы с Python благодаря различным плагинам.

Anaconda. Дистрибутив Python, который включает в себя среду разработки Spyder и Jupyter Notebook, предназначенный для научных вычислений и анализа данных.

IDLE. Официальная интегрированная среда разработки для Python, которая поставляется с пакетом установки Python и доступна для использования после установки интерпретатора.

В диссертационной работе используется облачный IDE от Google, а именно Google Colab (или Google Colaboratory), так как он имеет ряд преимуществ.

Google Colab позволяет пользователям создавать и выполнять код на языке Python через веб-браузер. Кроме того Google Colab предоставляет доступ к вычислительной мощности Google Cloud, что позволяет выполнять

сложные задачи, обучать модели машинного обучения, анализировать данные и многое другое, при этом без необходимости установки и настройки локальных сред разработки на компьютере. Это позволяет работать с программным кодом с любого устройства, что значительно увеличивает доступность и сохранность работы.

2.4 Решение тестируемой задачи с помощью генетического алгоритма

Генетический алгоритм для решения рассматриваемой задачи оптимизации может быть преставлен поэтапно:

- Определение структуры хромосомы. В данном случае, хромосома будет состоять из двадцати генов, каждый из которых представляет выбранный участок для соответствующего проекта. Ген будет принимать значения 0 или 1, то есть иметь вид $x_{ij} \in \{0; 1\}$ ($i = \underline{1,5}\,; j = \underline{1,4}$);
- Создание начальной популяции. Начальная популяция генерируется случайным образом и содержит набор случайных хромосом. Количество хромосом в популяции выберем равным 100;
- Оценка приспособленности (fitness). Для определения приспособленности каждой хромосомы, необходимо произвести расчет общего дохода (1), полученного при продаже зданий после реализации всех проектов, с учетом всех ранее описанных ограничений (2)-(5). Более приспособленная хромосома будет иметь большее значение;
- Селекция. Селекцию можно произвести на основе турнирного отбора;
- Кроссовер. В данном случае, для кроссовера можно использовать одноточечный метод, который разбивает хромосому на две части и меняет их между родительскими хромосомами;

– Мутация. Мутация может быть произведена случайной заменой одного гена на другой или изменением значения гена;
– Создание новой популяции. Новая популяция состоит из потомков, полученных в результате кроссовера и мутации;
– Повторять шаги 3-7 до достижения критерия остановки. Критерием остановки может быть достижение заданного количества итераций или достижение определенного уровня приспособленности;
– Выбор лучшей хромосомы. После завершения алгоритма, выбирается хромосома с наивысшей приспособленностью, которая и является решением задачи оптимизации.

Для получения решения рассматриваемой задачи оптимизации выполним реализацию генетического алгоритма на языке программирования Python.

Определим глобальные переменные которые будут использоваться в программе. Одна из используемых констант – это длина подлежащей оптимизации битовой строки [12],[23]. Длину битовой строки приравняем размерности задачи, т.е в нашем случае (5× 4) длина равна двадцати. Другие константы, обозначающие количество индивидуумов в популяции, вероятность скрещивания, вероятность мутации индивидуума, максимальное количество поколений, являющиеся константами генетического алгоритма, будем подбирать в ручную. Введем массивы с данными из таблиц 1-2. На рисунке 16 представленна часть кода, где вводятся глобальные переменные.

```
import random
import matplotlib.pyplot as plt
import math

# константы задачи
ONE_MAX_LENGTH = 20     # длина подлежащей оптимизации битовой строки

# константы генетического алгоритма
POPULATION_SIZE = 250   # количество индивидуумов в популяции
P_CROSSOVER = 0.9       # вероятность скрещивания
P_MUTATION = 0.1        # вероятность мутации индивидуума
MAX_GENERATIONS = 100   # максимальное количество поколений

RANDOM_SEED = 42
random.seed(RANDOM_SEED)
```

Рисунок 16 – Глобальные переменные

```
income=[7, 30, 10, 16, 29, 14, 28, 21, 16, 23, 29, 12, 8, 24, 9, 12, 24, 27, 13, 11 ]
cost=[23, 48, 48, 43, 45, 31, 36, 35, 46, 50, 38, 45, 47, 22, 41, 23, 39, 21, 32, 45 ]
```

Рисунок 17 – Доход и затраты

На рисунке 17 представленна часть кода, в которой вносяться таблицы расхода и затрат. Для вычисления приспособленности индивидуума воспользуемся функцией (1). Составим функцию для турнирного отбора и функцию для одноточеного скрещевания, а так же функцию, которая позволяет мутировать особям[13]. Мутирование происходит с ранее определенной вероятностью, в результате мутирования меняется значение одного из генов.

После всех перечисленных операций перейдем к написанию кода в соответствие с алгоритмом запускающий функции. На рисунке 18 17 представленна часть кода, в которой написана функция турнирного отобора и функция скрещевания.

```
def selTournament(population, p_len):
    offspring = []
    for n in range(p_len):
        i1 = i2 = i3 = 0
        while i1 == i2 or i1 == i3 or i2 == i3:
            i1, i2, i3 = random.randint(0, p_len-1), random.randint(0, p_len-1), random.randint(0, p_len-1)

        offspring.append(max([population[i1], population[i2], population[i3]], key=lambda ind: ind.fitness.values[0]))

    return offspring

def cxOnePoint(child1, child2):
    s = random.randint(2, len(child1)-3)
    child1[s:], child2[s:] = child2[s:], child1[s:]
```

Рисунок 18 – Функции турнирного отбора и скрещевание

На рисунке 19 представлена функция футация изменяющая в случайном порядке один ген с единице на ноль или с нуля на единицу.

```
def mutFlipBit(mutant, indpb=0.01):
    for indx in range(len(mutant)):
        if random.random() < indpb:
            mutant[indx] = 0 if mutant[indx] == 1 else 1
```

Рисунок 19 – Функция мутации

```
def oneMaxFitness(individual):
    COST=[cost[x] for x in range(ONE_MAX_LENGTH) if individual[x]==1]
    INC=[income[x] for x in range(ONE_MAX_LENGTH) if individual[x]==1]
    I=sum(INC)
    ogranich1=0
    indiv12=0
    number=0
    OGR2=0
    for i in range(len (individual)):
      number=number+1
      indiv12=individual[i]
      ogranich1=ogranich1+indiv12
      if number%4==0:
        if ogranich1==1:
          ogranich1=0
          OGR2=OGR2+1
          if OGR2==5:
            if sum(COST)>=200:
              return 0,
            else:
              return float(I/10),
        else:
          return 0,
```

Рисунок 20 – Функция приспособленности

После выполнения алгоритма необходимо вывести на экран данные, полученные в ходе его работы любым удобным способом [28]. В нашем случае вывод представляет собой вывод индивидуума с максимальной приспособленностью в поколении, а также результат средней приспособленности поколения.

Глава 3 Сравнительный анализ результатов работы программы

3.1 Результаты работы программы

Результаты работы программы представим выводом на экран лучшего индивидуума, то есть индивиидума с максимальной приспособленостью, а также самого значения максимальной приспасобленности и значения средней приспособленности поколения.

Для первого поколения получено:

- максимальная приспособленость равна 0,
- средняя приспособленость тоже равна 0,
- суммарные затраты равны 263,
- лучший индивидуум является строкой 0 1 1 0 0 0 1 1 0 1 1 0 0 1 0 1 1 1 0 0.

Полученный результат внесем в таблицу 9.

Таблица 9 – Лучший индивидуум первого поколения

	A	B	C	D
1	0	1	1	0
2	0	0	1	1
3	0	1	1	0
4	0	1	0	1
5	1	1	0	0

На рисунке 21 представлены результаты программы с первого по восьмое поколение.

```
Поколение 1: Макс приспособ. = 0, Средняя приспособ.= 0.0, Затраты.= 263
Лучший индивидуум =  0 1 1 0 0 0 1 1 0 1 1 0 0 1 0 1 1 1 0 0

Поколение 2: Макс приспособ. = 0, Средняя приспособ.= 0.0, Затраты.= 379
Лучший индивидуум =  0 0 0 0 1 1 1 1 1 1 0 0 1 1 1 1 1 0 0 0

Поколение 3: Макс приспособ. = 0, Средняя приспособ.= 0.0, Затраты.= 350
Лучший индивидуум =  1 0 0 1 1 1 0 0 0 0 1 0 1 0 1 0 1 1 0 0

Поколение 4: Макс приспособ. = 9.9, Средняя приспособ.= 0.0396, Затраты.= 446
Лучший индивидуум =  0 0 0 1 0 0 0 1 0 1 0 0 0 0 0 1 0 1 0 0

Поколение 5: Макс приспособ. = 0, Средняя приспособ.= 0.0, Затраты.= 428
Лучший индивидуум =  1 0 1 1 1 0 0 1 0 0 1 0 0 1 1 1 0 1 0 1

Поколение 6: Макс приспособ. = 0, Средняя приспособ.= 0.0, Затраты.= 332
Лучший индивидуум =  0 0 1 0 1 0 1 0 0 1 0 0 1 0 0 1 1 0 1 0

Поколение 7: Макс приспособ. = 10.9, Средняя приспособ.= 0.078, Затраты.= 507
Лучший индивидуум =  0 0 0 1 1 0 0 0 0 0 1 0 1 0 0 0 0 1 0 0

Поколение 8: Макс приспособ. = 8.6, Средняя приспособ.= 0.0344, Затраты.= 368
Лучший индивидуум =  1 0 0 0 0 0 0 1 0 1 0 0 0 1 0 0 0 0 0 1
```

Рисунок 21 – Результат 1-8 поколений

Максимальная приспособленность, равная 0, показывает, что лучший индивидуум первого поколения не является искомым решением.

С каждым поколением происходит увеличение приспособленности, в следствие чего, приближение к оптимальному решению задачи.

Для пятидесятого поколения получено:

– максимальная приспособленость равна 101,

– средняя приспособленость близка к 85,

– суммарные затраты равны 147,

– лучший индивидуум является строкой 0 0 0 1 1 0 0 0 0 0 1 0 0 0 0 0 1 0 0

Полученный результат оформим в виде таблицы 10.

Таблица 10 – Лучший индивидуум пятедесятого поколения

	A	B	C	D
1	0	0	0	1
2	1	0	0	0
3	0	0	1	0
4	0	0	0	0
5	0	1	0	0

На рисунке 22 представены результаты программы на 47-56 поколений

```
Поколение 47: Макс приспособ. = 11.6, Средняя приспособ.= 10.671999999999954, Затраты.= 149
Лучший индивидуум =  1 0 0 0 1 0 0 0 0 0 1 0 0 1 0 0 0 1 0 0

Поколение 48: Макс приспособ. = 12.5, Средняя приспособ.= 10.812399999999954, Затраты.= 149
Лучший индивидуум =  0 0 0 1 1 0 0 0 0 0 1 0 0 1 0 0 0 1 0 0

Поколение 49: Макс приспособ. = 12.5, Средняя приспособ.= 10.818399999999954, Затраты.= 149
Лучший индивидуум =  0 0 0 1 1 0 0 0 0 0 1 0 0 1 0 0 0 1 0 0

Поколение 50: Макс приспособ. = 12.5, Средняя приспособ.= 10.839999999999954, Затраты.= 149
Лучший индивидуум =  0 0 0 1 1 0 0 0 0 0 1 0 0 1 0 0 0 1 0 0

Поколение 51: Макс приспособ. = 12.5, Средняя приспособ.= 10.63679999999996, Затраты.= 149
Лучший индивидуум =  0 0 0 1 1 0 0 0 0 0 1 0 0 1 0 0 0 1 0 0

Поколение 52: Макс приспособ. = 12.5, Средняя приспособ.= 10.772399999999964, Затраты.= 169
Лучший индивидуум =  0 0 0 1 1 0 0 0 0 0 1 0 0 1 0 0 0 1 0 0

Поколение 53: Макс приспособ. = 12.5, Средняя приспособ.= 11.035599999999976, Затраты.= 149
Лучший индивидуум =  0 0 0 1 1 0 0 0 0 0 1 0 0 1 0 0 0 1 0 0

Поколение 54: Макс приспособ. = 12.5, Средняя приспособ.= 11.45919999999999, Затраты.= 149
Лучший индивидуум =  0 0 0 1 1 0 0 0 0 0 1 0 0 1 0 0 0 1 0 0

Поколение 55: Макс приспособ. = 12.5, Средняя приспособ.= 11.534399999999998, Затраты.= 169
Лучший индивидуум =  0 0 0 1 1 0 0 0 0 0 1 0 0 1 0 0 0 1 0 0

Поколение 56: Макс приспособ. = 12.5, Средняя приспособ.= 11.592799999999999, Затраты.= 169
Лучший индивидуум =  0 0 0 1 1 0 0 0 0 0 1 0 0 1 0 0 0 1 0 0
```

Рисунок 22 – результат 47-56 поколений

Максимальная приспособленность не равна нулю, данный индивидуум подходит под ограничения. Далее проверим является ли полученное решение самым лучшим. Рассмотрим следующие поколения.

Уже на 71 поколении можно увидеть улучшение решения. В этом поколении:

– максимальная приспособленость равна 104,
– средняя приспособленость около 92,
– суммарные затраты равны 160,
– лучший индивидуум является строкой 0 1 0 0 0 0 0 1 0 0 1 0 0 0 0 0 1 0 0 0.

Полученный результат оформим в виде таблицы 11.

Таблица 11 – Лучший индивидуум семьдясят первого поколения

	A	B	C	D
1	0	1	0	0
2	0	0	0	1
3	0	0	1	0
4	0	0	0	0
5	1	0	0	0

Максимальная приспособленность увеличилась и не равна нулю по этому, данный индивидуум не только подходит под ограничения, но и стал лучше.

На рисунке 23 представены результаты программы на 62-72 поколений

```
Поколение 62: Макс приспособ. = 12.5, Средняя приспособ.= 11.6, Затраты.= 192
Лучший индивидуум =  0 0 0 1 1 0 0 0 0 0 1 0 0 1 0 0 0 1 0 0

Поколение 63: Макс приспособ. = 12.5, Средняя приспособ.= 11.6448, Затраты.= 169
Лучший индивидуум =  0 0 0 1 1 0 0 0 0 0 1 0 0 1 0 0 0 1 0 0

Поколение 64: Макс приспособ. = 13.9, Средняя приспособ.= 11.6496, Затраты.= 169
Лучший индивидуум =  0 1 0 0 1 0 0 0 0 0 1 0 0 1 0 0 0 1 0 0

Поколение 65: Макс приспособ. = 13.9, Средняя приспособ.= 11.8612, Затраты.= 169
Лучший индивидуум =  0 1 0 0 1 0 0 0 0 0 1 0 0 1 0 0 0 1 0 0

Поколение 66: Макс приспособ. = 13.9, Средняя приспособ.= 11.7392, Затраты.= 169
Лучший индивидуум =  0 1 0 0 1 0 0 0 0 0 1 0 0 1 0 0 0 1 0 0

Поколение 67: Макс приспособ. = 13.9, Средняя приспособ.= 11.633600000000005, Затраты.= 169
Лучший индивидуум =  0 1 0 0 1 0 0 0 0 0 1 0 0 1 0 0 0 1 0 0

Поколение 68: Макс приспособ. = 13.9, Средняя приспособ.= 11.458800000000007, Затраты.= 169
Лучший индивидуум =  0 1 0 0 1 0 0 0 0 0 1 0 0 1 0 0 0 1 0 0

Поколение 69: Макс приспособ. = 13.9, Средняя приспособ.= 11.664000000000014, Затраты.= 169
Лучший индивидуум =  0 1 0 0 1 0 0 0 0 0 1 0 0 1 0 0 0 1 0 0

Поколение 70: Макс приспособ. = 13.9, Средняя приспособ.= 11.939200000000026, Затраты.= 169
Лучший индивидуум =  0 1 0 0 1 0 0 0 0 0 1 0 0 1 0 0 0 1 0 0

Поколение 71: Макс приспособ. = 13.9, Средняя приспособ.= 12.714800000000047, Затраты.= 174
Лучший индивидуум =  0 1 0 0 1 0 0 0 0 0 1 0 0 1 0 0 0 1 0 0

Поколение 72: Макс приспособ. = 13.9, Средняя приспособ.= 12.44880000000005, Затраты.= 174
Лучший индивидуум =  0 1 0 0 1 0 0 0 0 0 1 0 0 1 0 0 0 1 0 0
```

Рисунок 23 – результат 62-72 поколений

Далее проверим нельзя ли улучшить полученное решение. Рассмотрим следующие поколения.

Рассмотрим сотое поколение, как окончательное поколение работы программы. В сотом поколении получили, что

- максимальная приспособленость равна 104,
- средняя приспособленость около 93,
- суммарные затраты равны 160.
- лучший индивидуум является строкой 0 1 0 0 0 0 0 1 0 0 1 0 0 0 0 0 1 0 0 0 0.

Полученный результат оформим в виде таблицы 12.

Таблица 12 – Лучший индивидуум сотого поколения

	A	B	C	D
1	0	1	0	0
2	0	0	0	1
3	0	0	1	0
4	0	0	0	0
5	1	0	0	0

Сравнивая результаты семьдесят первого и сотого поколений, заметим, что максимальная приспособленость и суммарные затраты не изменились, средняя приспособленость увеличилась не более, чем на единицу. Поэтому процесс не стоит больше продолжать. Можно считать, что получено оптимальное решение, лучшему индивидуму будет соответствовать строка: 0 1 0 0 0 0 0 1 0 0 1 0 0 0 0 0 1 0 0 0.

```
Поколение 90: Макс приспособ. = 13.9, Средняя приспособ.= 12.788000000000054, Затраты.= 174
Лучший индивидуум =  0 1 0 0 1 0 0 0 0 0 1 0 0 1 0 0 0 1 0 0

Поколение 91: Макс приспособ. = 13.9, Средняя приспособ.= 13.394000000000057, Затраты.= 174
Лучший индивидуум =  0 1 0 0 1 0 0 0 0 0 1 0 0 1 0 0 0 1 0 0

Поколение 92: Макс приспособ. = 13.9, Средняя приспособ.= 12.843600000000054, Затраты.= 174
Лучший индивидуум =  0 1 0 0 1 0 0 0 0 0 1 0 0 1 0 0 0 1 0 0

Поколение 93: Макс приспособ. = 13.9, Средняя приспособ.= 12.676800000000052, Затраты.= 174
Лучший индивидуум =  0 1 0 0 1 0 0 0 0 0 1 0 0 1 0 0 0 1 0 0

Поколение 94: Макс приспособ. = 13.9, Средняя приспособ.= 13.177200000000056, Затраты.= 174
Лучший индивидуум =  0 1 0 0 1 0 0 0 0 0 1 0 0 1 0 0 0 1 0 0

Поколение 95: Макс приспособ. = 13.9, Средняя приспособ.= 13.288000000000055, Затраты.= 174
Лучший индивидуум =  0 1 0 0 1 0 0 0 0 0 1 0 0 1 0 0 0 1 0 0

Поколение 96: Макс приспособ. = 13.9, Средняя приспособ.= 13.054800000000053, Затраты.= 174
Лучший индивидуум =  0 1 0 0 1 0 0 0 0 0 1 0 0 1 0 0 0 1 0 0

Поколение 97: Макс приспособ. = 13.9, Средняя приспособ.= 13.177200000000056, Затраты.= 174
Лучший индивидуум =  0 1 0 0 1 0 0 0 0 0 1 0 0 1 0 0 0 1 0 0

Поколение 98: Макс приспособ. = 13.9, Средняя приспособ.= 12.676800000000052, Затраты.= 212
Лучший индивидуум =  0 1 0 0 1 0 0 0 0 0 1 0 0 1 0 0 0 1 0 0

Поколение 99: Макс приспособ. = 13.9, Средняя приспособ.= 13.330800000000055, Затраты.= 174
Лучший индивидуум =  0 1 0 0 1 0 0 0 0 0 1 0 0 1 0 0 0 1 0 0

Поколение 100: Макс приспособ. = 13.9, Средняя приспособ.= 12.899200000000054, Затраты.= 174
Лучший индивидуум =  0 1 0 0 1 0 0 0 0 0 1 0 0 1 0 0 0 1 0 0
```

Рисунок 24 – результат 90-100 поколений

На рисунке 24 представены результаты программы на 90-100 поколений

Согласно таблице 12, для проекта A нужно выбрать участок 5, для проекта B нужно выбрать участок 1, для проекта C нужно выбрать участок 3, для проекта D нужно выбрать участок 2. В таком случае получаемый доход составит 104 д.е. , затраты равны 160 д.е. при бюджете 200 д.е.

3.2 Анализ зависимости максимальной и средней приспособленности от поколения

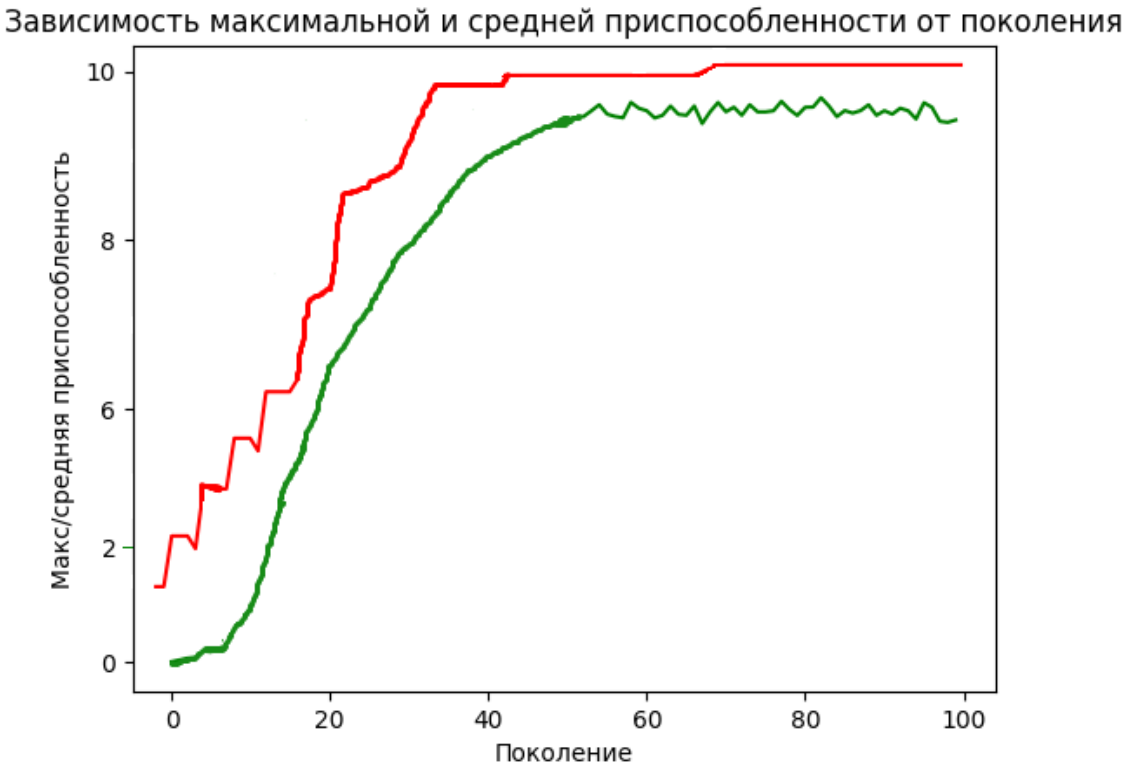


Рисунок 25 – График максимальной и средней приспособленности на 100 поколений

На рисунке 25 представлен график, показывающий максимальную и среднюю приспособленность в течение ста поколений. На графике красная

линия показывает максимальную приспособленность поколения, а зеленая обозначает среднюю приспособленность.

Из графика видно, что максимальная приспособленность достигла значения 35 и долго сохраняла это значение, но после 40 поколения произошел еще один скачок. С 40 по 70 поколение наблюдались никакие изменения, но в 71 поколении приспособленность снова начала расти. Максимальная приспособленность не менялась с 71 по 100 поколение, и значит, лучший индивидуум в этих поколениях является оптимальным для данной задачи.

Средняя же приспособленность постоянно менялась. До 50 поколения происходил рост, что обусловлено стремлением всех особей одного поколения к лучшим особям поколения. Однако с 50 поколения и далее роста средней приспособленности не наблюдается, а, наоборот, происходят колебания. Это связано с тем, что с каждым поколением некоторое количество особей мутируют, что не дает всему поколению приблизиться к лучшему индивидууму.

Заключение

В диссертационной работе анализируется математическая теория оптимизации, формулируется общая постановка задачи оптимизации, классифицируются математические методы оптимизации, приводятся прикладные возможности математических методов оптимизации.

Подробно рассматриваются два метода: симплексный метод и метод, созданный на основе искуственного интелекта, генетический алгоритм. Первый метод, при условии, что задача имеет решение, дает точный результат. С помощью второго метода можно получить только приближенное решение, но с меньшими затратами по времени и объему памяти.

В диссертационной работе разработан алгоритмический подход с использованием генетического алгоритма для решения линейной задачи оптимизации. Проанализированы возможности использования генетического алгоритма для решения задач оптимизации, изложены разновидности методов отбора, скрещевания и мутации в генетическом алгоритме.

Программная реализация генетического алгоритма осуществлена на основе пакетов Python. Программа успешно прошла тестирование. На основе программной реализации демонстрируется зависимость максимальной и средней приспособленности от поколения

Диссертационная работа в целом направлена на разработку и анализ алгоритмического подхода с использованием генетического алгоритма с целью повышения эффективности решения сложных задач оптимизации и поиска наилучших решений.

Список используемой литературы и используемых источников

1. Абдрахманов, Б. Т. Языки программирования высокого уровня / Б. Т. Абдрахманов // Актуальные научные исследования : Сборник научных трудов по материалам IV Международной междисциплинарной конференции, Москва, 28 июня 2022 года. – Москва: ОБЩЕСТВО С ОГРАНИЧЕННОЙ ОТВЕТСТВЕННОСТЬЮ "НАУЧНО-ИЗДАТЕЛЬСКИЙ ЦЕНТР ТОЛМАЧЕВО", 2022. – С. 14-17. – DOI 10.56545/9785604835760_14.

2. Аксёнов, Е.П. Методы оптимальных решений: учебное пособие / М-во с.х. РФ; федеральное гос. бюджетное образов.учреждение высшего образов. «Пермская гос. с.-х. акад. им.акад. Д.Н. Прянишникова». – Пермь : ИПЦ «Прокрость», 2016. – 90 с.

3. Аникина О.В., Гущина О.М. Использование технологий табличного моделирования генетических алгоритмов. / Прикладная информатика. Научные статьи, 2016, №3(63).

4. Борсук, Н. А. Сравнительный анализ алгоритмических языков высокого и низкого уровня / Н. А. Борсук, Д. В. Мастыкаш // Перспективы инновационных научно-практических исследований и разработок : сборник статей международной научной конференции, Санкт-Петербург, 24 февраля 2023 года. – Санкт-Петербург: Частное научно-образовательное учреждение дополнительного профессионального образования Гуманитарный национальный исследовательский институт «НАЦРАЗВИТИЕ», 2023. – С. 24-25.

5. Бураков М.В. Генетический алгоритм: теория и практика [Электронный ресурс] : учебное пособие / М.В. Бураков. - СПб. : ГУАП, 2008. – 164 с.

6. Вирсански Э. Генетические алгоритмы на a Python / пер. с английского А. А. Слинкина. – М.: ДМК Пресс, 2020. – 286 с.

7. Власова Е.А., Емельянова Н.З., Емельянова А.А. и др. Комплексное имитационное моделирование с применением генетических алгоритмов. / Прикладная информатика. Научные статьи, 2017, №6(72).

8. Генетический алгоритм: теория и практика: учеб. пособие / М. В. Бураков. – СПб.: ГУАП, 2008. – 164 с.: ил. ISBN 978-5-8088-0298-8.

9. Гладких Б.А. Методы оптимизации и исследования операций. Часть 1. Введение в исследование операций. Линейное программирование. — Томск: НТЛ, 2009. — 200 с.

10. Гладков Л.А., Курейчик В.В., Курейчик В.М. Генетические алгоритмы. М. : ФИЗМАТЛИТ, 2010. 368 с.

11. Дедов, С. В. Анализ преимуществ наиболее востребованных современных языков программирования / С. В. Дедов, О. Д. Кирсанов, О. Ю. Тимошевская // Актуальные вопросы современной науки : сборник статей по материалам XVII международной научно-практической конференции, Томск, 19 декабря 2018 года. Том Часть 1(4). – Томск: Общество с ограниченной ответственностью Дендра, 2018. – С. 63-72.

12. Коваленко, Т. А. Анализ языков программирования / Т. А. Коваленко, А. Ю. Лобачев // Фундаментальные проблемы основных направлений научно-технических исследований : сборник статей по итогам Международной научно-практической конференции, Волгоград, 17 марта 2018 года. – Волгоград: Общество с ограниченной ответственностью "Агентство международных исследований", 2018. – С. 48-50.

13. Кочегурова, Е. А. Теория и методы оптимизации : учебное пособие для вузов / Е. А. Кочегурова. — Москва : Издательство Юрайт, 2024. — 133 с. — (Высшее образование). — ISBN 978-5-534-10090-7.

14. Кудашов В.Н., Селина Е.Г., Основы линейного программирования– СПб: Университет ИТМО, 2020. – 42 с.

15. Курейчик В.М., Рокотянский А.А. Генетический алгоритм решения логистической задачи. // Известия ЮФУ. Технические науки. 2012. № 11. с. 245-251.

16. Марк Лутц. Программирование на Python / Пер. с англ. — 4-е изд. — СПб.: Символ-Плюс, 2011. — Т. I. — 992 с.

17. Методы оптимизации : учебное пособие / С.В. Каштаева; Министерство сельского хозяйства Российской Федерации, федеральное государственное бюджетное образовательное учреждение высшего образования «Пермский аграрно-технологический университет имени академика Д.Н. Прянишникова». – Пермь : ИПЦ «Прокростъ», 2020. – 84 с ; 21 см – Библиогр.: с.83. – 35 экз. – ISBN 978-5-94279-501-6 – Текст : непосредственный.

18. Панченко, Т. В. Генетические алгоритмы [Текст] : учебно-методическое пособие / под ред. Ю. Ю. Тарасевича. — Астрахань : Издательский дом «Астраханский университет», 2007. — 87 [3] с.

19. Певнева А.Г., Калинкина М.Е., Методы оптимизации– СПб: Университет ИТМО, 2020. – 64 с.

20. Профессиональное программирование на VBA в Excel 2002. : Пер. с англ.— М. : Издательский дом '"Вильяме", 2003. — 784 с. : ил. — Парал. тит. англ. ISBN 5-8459-0541-9 (рус.).

21. Соложенцева, Р. С. Применение языка программирования PYTHON в анализе данных / Р. С. Соложенцева // Вестник Университета управления "ТИСБИ". – 2021. – № 4. – С. 103-112.

22. Сопов Е.А. Эволюционные алгоритмы моделирования и оптимизации сложных систем. Красноярск, 2004.

23. Чернова, С. В. Сравнительный анализ языков программирования высокого уровня Python и C++ / С. В. Чернова, В. А. Ларина // Аспирант и соискатель. – 2019. – № 5(113). – С. 9-11.

24. An Introduction to Genetic Algorithms Mitchell Melanie A Bradford Book The MIT Press Cambridge, Massachusetts • London, England Fifth printing, 1999 First MIT Press paperback edition, 1998 Copyright © 1996 Massachusetts Institute of Technology.

25. Analysis of theoretical and methodological bases of teaching object - oriented programming languages in higher school / B. Shayakhmetova, N. Orumbayeva, Sh. Omarova, Yu. Antipov // Bulletin of the Karaganda University. Mathematics Series. – 2017. – No. 4(88). – P. 73-79. – DOI 10.31489/2017M4/73-79.

26. Badariah, Sh. Comparison between Genetic Algorithm and Genetic Programming Performance for Photomosaic Generation [Text] / Shahrul Badariah Mat Sah, Vic Ciesielski, Daryl D'Souza, Marsha Berry // AsiaPacific Conference on Simulated Evolution and Learning – 7th International Conference, SEAL 2008, Melbourne, Australia, December 7-10, 2008. Proceedings: Simulated Evolution and Learning. – Springer-Verlag Berlin Heidelberg 2008. – pp. 259-268.

27. Karthikeyan, P. Improvement in Genetic Algorithm with Genetic Operator Combination (GOC) and Immigrant Strategies for Multicast Routing in Ad Hoc Networks [Text] / P. Karthikeyan, Subramanian Baskar // International Conference on Swarm, Evolutionary, and Memetic Computing – 4th International Conference, SEMCCO 2013, Chennai, India, December 19-21, 2013, Proceedings, Part I: Swarm, Evolutionary, and Memetic Computing. – Springer International Publishing Switzerland 2013. – pp. 481-491.

28. Mzili, I. Hybrid Penguins Search Optimization Algorithm and Genetic Algorithm Solving Traveling Salesman Problem [Text] / Ilyass Mzili, Mohammed Essaid Riffi, Fatiha Benzekri /Advanced Information Technology, Services and Systems. – Springer International Publishing AG 2018. – pp. 461-473.

29. Oliveira, S. Genetic Seam Carving: A Genetic Algorithm Approach for Content-Aware Image Retargeting [Text] / Saulo A. F. Oliveira, Francisco N. Bezerra, Ajalmar R. Rocha Neto / Proceedings: Pattern Recognition and Image Analysis. – Springer International Publishing Switzerland 2015. – pp. 700- 707.

30. Xia, Yi. An Improved FastSLAM Algorithm Based on Genetic Algorithms [Text] / Yi-min Xia, Yi-min Yang / Revised Selected Papers: Information and Automation. – Springer-Verlag Berlin Heidelberg 2011. – pp. 296-302.