Мухамедиева Дилноз Тулкуновна

РЕШЕНИЕ ЗАДАЧ ОПТИМИЗАЦИИ НА ОСНОВЕ КВАНТОВЫХ ВЫЧИСЛЕНИЙ

Монография

Мухамедиева Дилноз Тулкуновна

РЕШЕНИЕ ЗАДАЧ ОПТИМИЗАЦИИ НА ОСНОВЕ КВАНТОВЫХ ВЫЧИСЛЕНИЙ

Монография

Национальный исследовательский университет «Ташкентский институт инженеров ирригации и механизации сельского хозяйства»

Мухамедиева Дилноз Тулкуновна

РЕШЕНИЕ ЗАДАЧ ОПТИМИЗАЦИИ НА ОСНОВЕ КВАНТОВЫХ ВЫЧИСЛЕНИЙ

Ташкент-2024

Издательство «Fan ziyosi»

УДК 519.71(575.1)
Д.Т.Мухамедиева. «Решение задач оптимизации на основе квантовых вычислений». Монография – Т.: Изд.«Fan ziyosi», 2024. 346 с.

В работе рассмотрены актуальные теоретико-методические проблемы применения технологии искусственного интеллекта и перспективы квантовых вычислений для решения задач оптимизации. Квантовая теория не только расширяет нашу научную картину мира, но и вносит глубокие изменения в наши обыденные представления о реальности. Она показывает, что мир намного более сложен и удивителен, чем мы могли предполагать, и предоставляет инструменты для исследования этой удивительной сложности. В связи с нетрадиционностью применяемого математического аппарата в книге приводятся основные положения теории квантовых вычислений в объеме, необходимом для понимания постановок задач, рассмотренных в последующих главах.

Рекомендовано к печати НТС
Национально-исследовательского университета «Ташкентский институт инженеров ирригации и механизации сельского хозяйства»

Рецензенты:

Маматов Н.С. - д.т.н., профессор Национально- исследовательского университета «Ташкентский институт инженеров ирригации и механизации сельского хозяйства».

Рахимов Н.О. – д.т.н., профессор Ташкентского университета информационных технологий.

С О Д Е Р Ж А Н И Е

Введение…… 5
Глава 1. КВАНТОВЫЕ ВЫЧИСЛЕНИЯ И ИНТЕЛЛЕКТУАЛЬНЫЙ АНАЛИЗ ДАННЫХ…… 7
1.1.Эра квантовых вычислений …….. 7
1.2.Данные. Подходы и определения……. 24
1.3.Большие данные.Системы управления Большими данными.. 33
Глава 2. КВАНТОВЫЕ ВЫЧИСЛЕНИЯ……... 40
2.1.Системы управления Большими данными …….. 40
2.2. Гейты и квантовые схемы……. 54
2.3. Квантовая схемотехника…… 71
2.4.Принципы квантовых вычислений……. 76
Глава 3. ПОДХОДЫ К РЕШЕНИЮ ЗАДАЧ ОПТИМИЗАЦИИ. 90
3.1. Анализ решения задач оптимизации классическими методами……. 90
3.2. Анализ решения задач с помощью метода случайного поиска…… 113
3.3. Анализ решения задач оптимизации эволюционными методами……... 126
Глава 4. РЕШЕНИЕ МНОГОКРИТЕРИАЛЬНЫХ ЗАДАЧ ОПТИМИЗАЦИИ НА ОСНОВЕ ИСПОЛЬЗОВАНИЯ ГЕНЕТИЧЕСКОГО АЛГОРИТМА…… 136
4.1. Анализ особенности генетических алгоритмов……. 136
4.2. Исследование сходимости генетических алгоритмов к глобальному оптимуму…… 150
4.3. Применение генетических алгоритмов к решению задач нахождения глобального оптимума…….. 161
4.4. Применение нечетких генетических алгоритмов с искусственным отбором для решения задач оптимизации…….. 174
4.5. Применение искусственных иммунных систем к решению задачи оценки состояния сложных процессов……. 183

4.6. Применение природных вычислений к решению задач оптимизации…….. 192
4.7. Оптимизация многоэкстремальных функций с помощью генетических алгоритмов…… 205
4.8. Исследование характеристик муравьиных алгоритмов……. 245
Глава 5. АЛГОРИТМЫ КВАНТОВОЙ ОПТИМИЗАЦИИ…….. 250
5.1.Решение задачи нелинейной оптимизации на основе алгоритма Гровера…… 250
5.2. Модель биоразнообразия и устойчивости растений на основе квантовой вариационной оптимизации……. 277
5.3.Модель для лесных экосистем на основе квантовой оптимизации…….. 286
5.3. Variational Quantum Eigensolver метод для решения задач оптимизации в энергетических системах…… 296
Заключение……... 329
Список использованной литературы…… 330

Введение

Квантовые технологии, основанные на управлении индивидуальными квантовыми свойствами частиц, активно развиваются по всему миру. В технологически развитых странах в США, Китае, Канаде, ЕС, Великобритании, Японии, Австра лиии т. д. исследования и разработки в области квантовой физики находятся под бдительным вниманием со стороны государства: для их развития создаются специализированные центры компетен ций, а финансирование обеспечивается специаль ными целевыми программами. На сегодняшний день главным потребителем квантовых технологий является государство. Во многом это объясняется стратегической важностью квантовых техно логий для обеспечения защищенности интересов государства, например, для обеспечения безопас ности в информационной сфере. В Евросоюзе создана специальная программа quantum Flagship – после завершения предыдущей программы по развитию квантовых технологий. В Китае запущена программа по квантовым технологиям. Конгрессом США утверждена долгосрочная программа развития квантовых технологий National Quantum Initiative. Аналогичные программы рассматриваются в Великобритании, Японии, Канаде, Австралии и ряде других стран.

Квантовые технологии одно из наиболее динамически развивающихся направлений. Квантовые технологии открывают новые возможности для целого ряда областей. За счет своих уникальных свойств квантовые системы могут стать основой нового поколения высокопроизводительных вычислительных устройств квантовых компьютеров, методов защиты информации с использованием квантовой криптографии, а также высокоточных измерительных устройств квантовых сенсоров и квантовых метрологических устройств. Обзор посвящен прогрессу, наблюдаемому в основных сферах современных квантовых технологий: квантовой обработки информации, квантовой криптографии, а также квантовой метрологии и квантовой сенсорики.

С каждым годом вычислительные устройства становятся как более компактными, так и более производительными. то же движет столь стремительным прогрес сом вычислительных технологий? Основным стимулирующим фактором развития компьютеров считается совершенствование технологий, связанных с миниатюризацией элементной базы. Иными словами, мощность компьютеров растет, так как в каждом новом поколении компьютеров на чипе той же площади можно разместить примерно вдвое больше транзисторов. Это эмпирическое правило, известное сегодня как закон Мура, с достаточной степенью точности описывало развитие информационных технологий. Чтобы поддерживать дальнейший рост производительности компьютеров, необходимо будет создавать транзисторы атомных размеров. Стоит отметить, что рост других параметров, которые сопутствуют развитию вычислительных технологий, таких как тактовая частота процессоров, уже завершился. Таким образом, на этот раз развитие компьютеров сталкивается с новой физической парадигмой: не с привычной классической физикой, а с квантовой механикой.

Строго говоря, появление таких технологий, как транзисторы и лазеры также явилось результатом исследований в области квантовой физики. Однако в случае рассмотрения принципов функционирования лазера или транзистора речь идет о коллективных квантовых явлениях, проявляющихся на уровне коллектива большого числа квантовых объектов атомов, фотонов или электронов. Задача же управления индивидуальными кванто выми объектами, такими как одиночные фотоны, атомы, ионы оказывается гораздо сложнее: она находится на переднем крае развития науки. Для ее решения в случае рассмотрения сложных квантовых систем на уровне индивидуальных частиц нужно разработать эффективные методы создания, контроля и измерения. Отметим, что Нобелевская премия по физике была вручена С. Арошу и Д. Вайленду с формулировкой: За создание прорывных технологий манипулирования квантовыми системами, которые сделали возможными измерение отдельных квантовых систем и управление ими.

Глава 1. КВАНТОВЫЕ ВЫЧИСЛЕНИЯ И ИНТЕЛЛЕКТУАЛЬНЫЙ АНАЛИЗ ДАННЫХ

1.1.Эра квантовых вычислений

Квантовые вычисления начали развиваться более 40 лет назад [1,2] и до сегодняшнего дня остаются областью информационных технологий, привлекающей внимание ученых из разных стран [3, 4]. Благодаря способности квантовых компьютеров решать определенные вычислительные задачи экспоненциально быстрее классических, они могут найти применение в самых различных областях, таких как криптография, машинное обучение, искусственный интеллект, биология и т.д. Квантовые вычисления также могут быть использованы для решения вычислительных задач ВКО, таких как нечеткий поиск, оптимальное распределение информационных ресурсов, распознавание целей. Несмотря на то что создание полноценного квантового компьютера является на данный момент нерешенной задачей, разработка программных моделей квантовых вычислений остается актуальным направлением развития области. Такие программы используются для моделирования работы квантовых схем, а также изучения и поиска путей решения проблем, возникающих при реализации квантовых вычислений

Все разработанные и разрабатываемые программные модели квантовых вычислений оперируют такими базовыми понятиями, как кубит, квантовый регистр, гейт, квантовая схема. Одиночный кубит является суперпозицией двух квантовых состояний $|0\rangle$ и $|1\rangle$, каждое из которых может рассматриваться как носитель одного бита классической информации. Кубит представляется вектором единичной длины в трехмерном пространстве. Такое геометрическое изображение кубита называется его представлением на сфере Блоха.

Квантовый регистр шириной N состоит из N упорядоченных кубитов. Он может находиться в одном из двух состояний: измеренном или неизмеренном. Если квантовый регистр измерен, каждый кубит становится обычным битом, а квантовый регистр, соответственно, обычным регистром из N битов. Если квантовый регистр не измерен, то

имеет место квантовая суперпозиция и значением квантового регистра является информация о вероятностях «выпадения» при измерении каждого из 2^N возможных N-разрядных двоичных чисел, то есть квантовый регистр из N кубитов – это вектор длиной 2^N. Состояние квантового регистра из N кубитов представляет собой тензорное произведение вектор-столбцов состояний соответствующих кубитов.

Квантовый гейт – это матрица размером $2^m \times 2^m$, где m – количество кубитов, на которые воздействует гейт. Воздействие гейта на кубит выражается скалярным произведением.

В теории квантовых вычислений доказывается утверждение о том, что однокубитные гейты и гейт CNOT являются универсальными.

Квантовая схема – это последовательность преобразований из конечного набора гейтов. На вход квантовая схема получает квантовые биты. Результат ее работы вероятностный.

Идея квантовых компьютеров поражает своей потенциальной мощью и эффективностью. В отличие от классических компьютеров, которые используют биты для представления и обработки информации, квантовые компьютеры используют квантовые биты или кубиты, которые могут находиться в суперпозиции состояний и использовать квантовые явления, такие как квантовые перепутанные состояния и квантовые параллельные вычисления. Из-за этой особенности квантовые компьютеры способны обрабатывать огромные объемы информации с невероятной скоростью и эффективностью. Даже небольшое количество кубитов в квантовом компьютере может представлять огромный информационный ресурс, превосходящий число частиц во Вселенной. Однако важно отметить, что квантовые компьютеры находятся пока еще в стадии разработки и экспериментов, и существует ряд технических и алгоритмических проблем, которые необходимо решить, прежде чем они станут практически доступными для широкого использования. Но потенциал квантовых компьютеров внушает надежду на возникновение совершенно новых возможностей в области вычислений и информационных технологий.

Квантовая теория проникает гораздо глубже, чем просто в мир элементарных частиц. Она пересекает границы микромира и макромира, предоставляя новые инсайты в природу реальности. Одно из наиболее

захватывающих аспектов квантовой теории - это её способность менять наше представление о мире в целом. До открытия квантовой теории физики считали, что макромир подчиняется классическим законам механики, таким как законы Ньютона. Однако квантовая теория показала, что мир на самом деле функционирует на уровне, когда классические понятия перестают работать. Например, частицы могут находиться в состояниях суперпозиции, где они существуют в нескольких состояниях одновременно, или проявлять явления взаимозависимости, такие как квантовая запутанность. Эти квантовые явления, хотя и наблюдаются на микроуровне, начинают оказывать влияние на мир в целом. Например, они могут быть основой для новых технологий, таких как квантовые компьютеры, которые обещают революционизировать вычислительные возможности. Кроме того, они помогают нам лучше понять природу времени, пространства и информации.

Квантовая теория не только расширяет нашу научную картину мира, но и вносит глубокие изменения в наши обыденные представления о реальности. Она показывает, что мир намного более сложен и удивителен, чем мы могли предполагать, и предоставляет инструменты для исследования этой удивительной сложности. Квантовая механика применима не только к микроскопическим системам, но и к макроскопическим объектам. Однако в повседневной жизни мы обычно не замечаем квантовые эффекты из-за их малых масштабов или потому, что они редко проявляются в макромире. Когда масштабы увеличиваются, квантовые эффекты сглаживаются и кажутся незаметными в сравнении с классическими явлениями. Однако это не означает, что квантовые законы перестают действовать на макроуровне. Напротив, они остаются всеобщими и применимыми ко всему миру. В большинстве случаев для описания поведения макрообъектов можно использовать классическое приближение квантовой теории, которое учитывает квантовые эффекты в виде статистических закономерностей или вероятностных распределений. Тем не менее, есть исключения, когда квантовые эффекты становятся заметными на макроскопическом уровне. Например, это происходит в явлениях, таких как сверхпроводимость и сверхпроводимость, где макроскопическое

квантовое поведение играет решающую роль. Поэтому понимание квантовой механики важно не только для изучения микромира, но и для полного понимания мира в целом, включая его макроскопические аспекты.

Квантовая теория открывает перед нами новую перспективу на природу реальности, которая намного более сложна, чем мы могли себе представить на основе классических представлений. Она переворачивает нашу интуицию о том, как работает мир, и показывает, что некоторые явления, которые кажутся невероятными или сверхъестественными, на самом деле являются естественными следствиями квантовой механики. Вместо того чтобы рассматривать макроскопические объекты как просто совокупность их микроскопических частей, квантовая теория населяет мир фундаментальной неопределенностью и вероятностными законами. Это означает, что даже при полном знании о состоянии всех его составляющих частей мы не можем предсказать с абсолютной точностью его поведение. Вместо этого мы можем только говорить о вероятностях различных исходов.

Квантовая теория меняет наше представление о том, как работает мир, и заставляет нас пересмотреть классическое представление о соотношении между частью и целым. Она показывает, что в квантовом мире эта связь гораздо более сложна и интригующа, чем мы предполагали ранее. В квантовой теории акцентируется важность рассмотрения системы как целого. Это отличается от классического подхода, который обычно начинается с анализа отдельных компонентов системы, чтобы потом составить общее представление о её функционировании.

В квантовой механике подходят к системе как к единому целому, где свойства и состояния частей могут быть взаимосвязаны и изменяться в зависимости от контекста системы в целом. Этот подход отражает особенности квантовой неопределённости и взаимозависимости частиц, которые являются ключевыми элементами квантовой теории. В квантовой механике путь от части к целому считается более продуктивным, чем обратное направление, потому что он учитывает сложные квантовые взаимодействия и возможные

суперпозиции состояний. Этот подход позволяет более полно и точно описывать поведение системы в рамках квантовой теории. Квантовая механика переосмыслила наше понимание физической реальности, предлагая более фундаментальный подход, основанный на понятии состояний системы. Вместо того чтобы рассматривать объекты как набор физических характеристик, квантовая теория сосредотачивается на состояниях, которые определяют эти характеристики. Это позволяет применять методы квантовой теории к широкому спектру систем, включая как микрочастицы, так и макроскопические объекты. В квантовой механике нет никакого разделения между микромиром и макромиром: принципы квантовой теории применимы ко всей Вселенной в целом. Этот подход открывает новые возможности для понимания фундаментальных процессов в природе, а также для развития новых технологий и приложений. Квантовая механика становится не просто теорией микромира, но универсальным инструментом для изучения и анализа различных систем и явлений, как на микро-, так и на макроскопических уровнях.

Термин "квантовый" не должен ассоциироваться исключительно с микроскопическими объектами или явлениями. Он представляет собой широкий и фундаментальный способ описания реальности, основанный на понятии состояния системы и принципах квантовой механики. Этот подход применим ко всему миру в целом, как к микро-, так и к макроскопическим объектам и процессам. Использование термина "квантовый" в широком контексте подчеркивает его универсальность и применимость ко всем аспектам физической реальности. В этом смысле квантовая теория становится основой для понимания мира в его самых разнообразных проявлениях и масштабах. Важно осознавать, что наше мировоззрение и представления о реальности формируются на основе того, что мы знаем и понимаем о мире вокруг нас. И, даже если мы не задумываемся об этом, наше поведение и решения все равно опираются на наши установки и представления о мире.

Классическая физика, с её представлением о материи, полях и частицах, долгое время служила основой для нашего понимания мира. Однако, как и любая другая научная теория, она имеет свои ограничения и не может объяснить все аспекты реальности. В этом

смысле квантовая теория открывает новые горизонты и предоставляет более глубокое понимание природы вселенной. Она показывает нам, что мир намного более сложен и изощрен, чем мы предполагали, и что наши представления о материи и её структуре могут быть преодолены или дополнены новыми фундаментальными концепциями. Таким образом, понимание квантовой теории может помочь нам расширить наше мировоззрение и воспринимать мир в его более глубоком и сложном аспекте, что, в конечном счете, может изменить наш взгляд на себя и нашу роль в этой вселенной. Наши жизненные ценности, приоритеты и стремления в значительной степени определяются нашим мировоззрением и пониманием мира вокруг нас. Если наше представление о реальности ограничено и упрощено, то и наши ценности и стремления могут быть ограничены этими представлениями.

Квантовая теория предлагает более глубокое и сложное понимание реальности, которое не всегда соответствует нашим обыденным представлениям. Её открытия и принципы могут вызвать у нас существенный пересмотр наших установок и ценностей, а также изменить наше восприятие мира и нашу роль в нём. Поэтому важно осознавать значение квантовой теории не только для науки, но и для нашей повседневной жизни и понимания себя в этом мире. Она может помочь нам расширить границы нашего восприятия и понимания, открывая новые возможности для личного развития и познания.

Основной вывод квантовой теории — что материя и физические поля не являются основой окружающего мира, а лишь частью более глубокой и сложной Квантовой Реальности — имеет далеко идущие последствия, которые мы еще только начинаем осознавать. Этот вывод подразумевает, что существует что-то гораздо более фундаментальное и непостижимое, чем материя, и что наш мир может быть гораздо более сложным и удивительным, чем мы когда-либо предполагали. Это вызывает фундаментальные вопросы о природе реальности, нашем месте в этой реальности и о смысле нашего существования. Поэтому изучение и понимание квантовой теории имеет огромное значение для нашего мировоззрения, нашего понимания мира и нашего места в нем. Она ставит перед нами вызов расширить границы нашего восприятия и

открыться новым возможностям для исследования и познания Квантовой Реальности.

На рынке доступно так много инструментов анализа больших данных, но все еще существует проблема более высокого ранга, которую не могут решить в оптимальное время даже самые совершенные классические компьютеры. Чтобы обеспечить огромные вычислительные возможности современному обычному компьютеру, родилась концепция квантовых вычислений. Вычислительная мощность компьютера зависит от количества используемых транзисторов, и согласно закону Мура сегодня эта мощность удваивается каждые два года. По состоянию на 2014 год коммерчески доступный процессор с наибольшим количеством транзисторов — 15-ядерный Xeon IvyBridge — EX с 4,2 миллиарда транзисторов, а в случае графических процессоров nvidia TESLA, в которых количество транзисторов не превышает 7 миллиардов, но для высокопроизводительных вычислений требуется еще больше.

67 лет назад, когда был разработан первый транзистор, никто не мог предсказать, какую роль компьютер будет играть в нашем обществе сегодня через Интернет. Интернет не является мгновенным успехом. Он проделал путь ок. 24 года. За этот короткий промежуток времени это затрагивает почти все аспекты жизни. Мы не можем представить себе банковскую систему, систему бронирования железнодорожных билетов, данные авиакомпаний, бизнес-данные и т. д. без централизованного хранилища данных в Интернете. Из-за быстрого использования Интернета через социальные сети, службы электронной почты, веб-инструменты связи, веб-конференции и т. д. объем данных в Интернете быстро увеличился. Данные, которые выходят за рамки емкости хранения и возможностей обработки классического компьютера, называются большими данными, и получение некоторой информации из большого количества данных является очень большой проблемой. ИТ-компании, такие как Facebook, Google, Amazon, Salesforce и т. д., управляют своими данными в крупных быстроэластичных центрах обработки данных. Эти организации также предоставляют по запросу различные услуги, такие как хранилище (SaaS), платформа (PaaS), приложения (AaaS) и т. д. на условиях аренды, называемые облачными

вычислениями. Облачная инфраструктура очень хороша для небольших организаций, и они обращаются к поставщикам облачных услуг. Из-за огромного увеличения объема данных в облаке ИТ-отрасли сталкиваются с очень серьезной проблемой анализа больших данных [1-5].

Аналитика больших данных позволяет получить представление об огромном наборе данных, получить бизнес-аналитику, найти закономерности, сделать выводы и сделать некоторые прогнозы. На рынке доступно так много инструментов анализа больших данных, но все еще существует проблема более высокого ранга, которую не могут решить в оптимальное время даже самые совершенные классические компьютеры. Для анализа больших данных нам нужна система с огромными возможностями обработки. Как мы знаем, скорость обработки обычного компьютера зависит от количества транзисторов, которые мы используем. Увеличивая количество транзисторов, мы можем увеличить производительность обработки или использовать HDFS и MapReduce для более быстрой обработки данных. Когда мы говорим о высокопроизводительных вычислениях, термин «квантовые вычисления» радикально изменил наше мышление и предоставил возможности обработки вычислений в порядке 2^n для ввода n кубитов. Перспективы квантовых компьютеров заключаются в том, что вычисления, на которые обычные компьютеры тратят часы, квантовые компьютеры могут выполнить за секунды[6-12].

Большие данные. Данные, которые выходят за рамки возможностей хранения и обработки классического компьютера, называются большими данными [3-17].

Источники генерации данных: данные социальных сетей, сенсорная сеть, поиск в Интернете, геномика, астрономия, данные авиакомпаний и т. д.

Типы данных:

Структурированные данные: данные заданного формата, адресная книга, каталоги продуктов, банковские переходы.

Неструктурированные данные: данные, которые не имеют заранее установленного формата. Фильмы, аудио, текстовые файлы, веб-страницы, компьютерные программы, социальные сети.

Полуструктурированные данные: неструктурированные данные, которые можно структурировать с помощью доступных описаний форматов [18-23].

Причины больших данных:

I. Недорогое хранилище для хранения данных, которые были удалены ранее.

II. Мощные многоядерные процессоры.

III. Низкая задержка возможна благодаря распределенным вычислениям: компьютерный кластер и сети соединены через высокоскоростную сеть.

IV. Виртуализация: разделяйте, агрегируйте, изолируйте ресурсы любого размера и динамически изменяйте их.

V. Доступное хранилище и вычисления с минимальными затратами рабочей силы через облака.

VI. Лучшее понимание распределения задач (карта сокращения), вычислительной архитектуры (Hadoop).

VII. Передовые аналитические методы (машинное обучение).

VIII. Управляемые платформы больших данных: поставщики облачных услуг.

IX. Программное обеспечение с открытым исходным кодом: открытое

стек, PostGresSQL.

X. 12 марта 2012 г. Обама объявил о выделении 200 миллионов долларов на исследования больших данных для NSF, NIH, DOE, DoD, DARPA и USGS (геологическая разведка).

Квантовые вычисления: Квантовый компьютер — это компьютер, который использует законы квантовой механики для выполнения вычислений. Он может решать быстрее, чем современный самый быстрый компьютер. Он обещает более мощные вычислительные возможности, чем любой обычный компьютер [34-35].

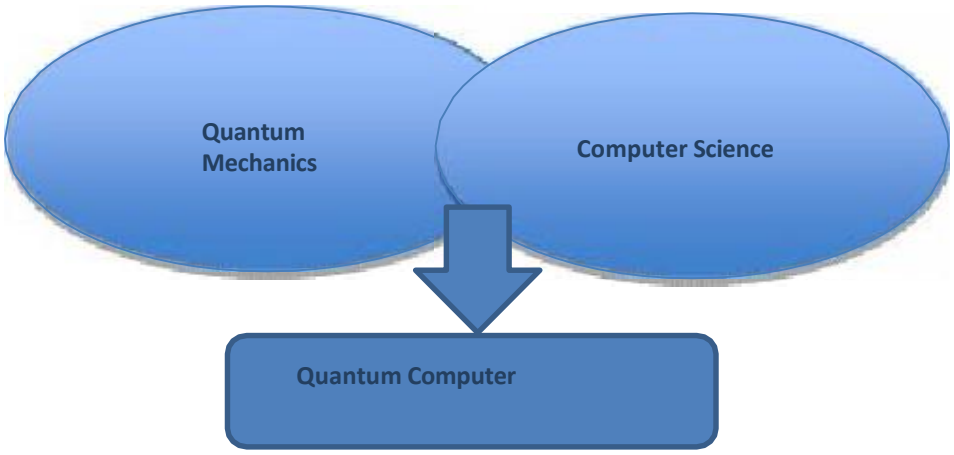


Рис. 1.1 – Связь квантового компьютера с различными областями

Кубит (квантовый бит). В квантовом компьютере мы используем кубит (битовый эквивалент обычного компьютера) для хранения данных [36].

Генерация кубита: Маленькие частицы, такие как электроны и фотоны, имеют спин, и этот спин можно измерить с помощью магнитного поля. Если мы поместим электрон в магнитное поле, то он будет вращаться в разных направлениях и одновременно, что называется квантовой суперпозицией. Если у нас есть n бит, то в результате их суперпозиции мы получим 2^n кубитов [37].

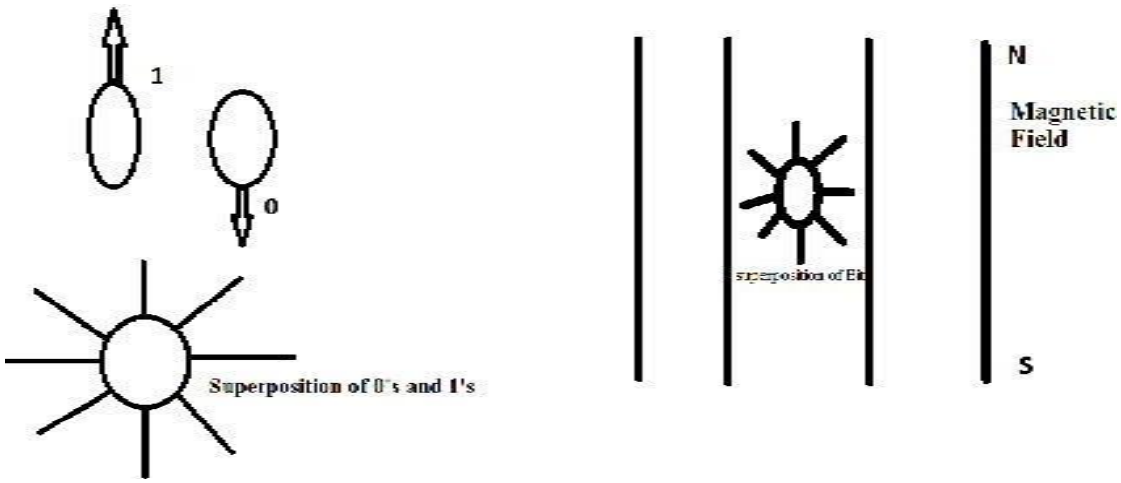


Рис. 1.2 –Генерация кубита

Кубит — это аналог бита для квантовых вычислений. Это означает, что кубит можно представить как линейную комбинацию |0> и |1>:

$$|\varphi\rangle = a|0\rangle + b|1\rangle,$$

где a и b — амплитуды вероятности и комплексные числа. Когда мы измеряем этот кубит, вероятность результата |0> равна $|a|^2$, а вероятность результата |1> равна $|b|^2$. Поскольку абсолютные квадраты амплитуд равны вероятностям, из этого следует, что a и b должны быть ограничены уравнением

$$|a|^2 + |b|^2 = 1.$$

То есть мы должны измерять либо одно состояние, либо другое.

Сфера Блоха: кубит $|\varphi\rangle = a|0\rangle + b|1\rangle$ можно представить в виде точки (θ, φ) на единичной сфере, называемой сферой Блоха. Определите углы θ и φ, разрешив $a = \cos\left(\theta/2\right)$ и $b = e^{i\varphi}\sin\left(\theta/2\right)$. Здесь a считается действительным, что всегда можно сделать реальным, умножив на $|\psi\rangle$ общий фазовый коэффициент (который ненаблюдаемый). Тогда $|\psi\rangle$ представляется единым вектором, называемый вектором Блоха.

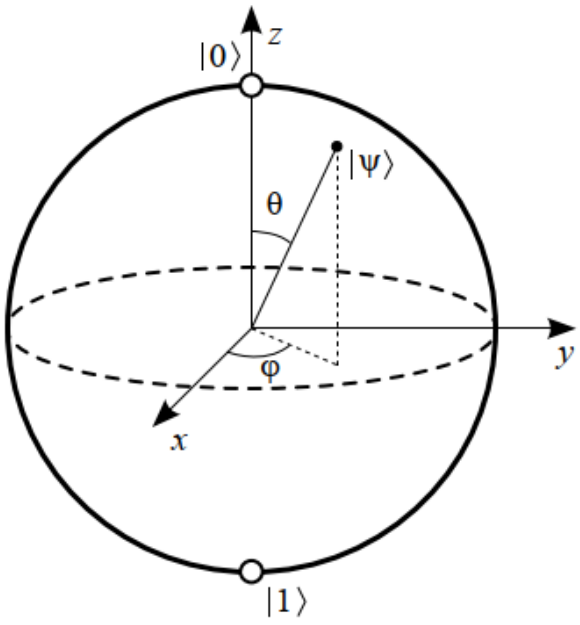


Рис. 1.3 – Сфера Блоха

Квантовая запутанность: два объекта, если они квантово-механически запутаны, то они сильно связаны друг с другом, даже если они находятся на огромном расстоянии друг от друга. Это значит, что суперпозиция битов и все одновременно. Электроны внутри атома существуют на квантованных энергетических уровнях. Качественно эти электронные орбиты можно рассматривать как резонирующие стоячие волны, что находится в тесной аналогии с вибрирующими волнами, которые можно наблюдать на туго удерживаемой струне. Два таких отдельных уровня могут быть изолированы для настройки базовых состояний кубита.

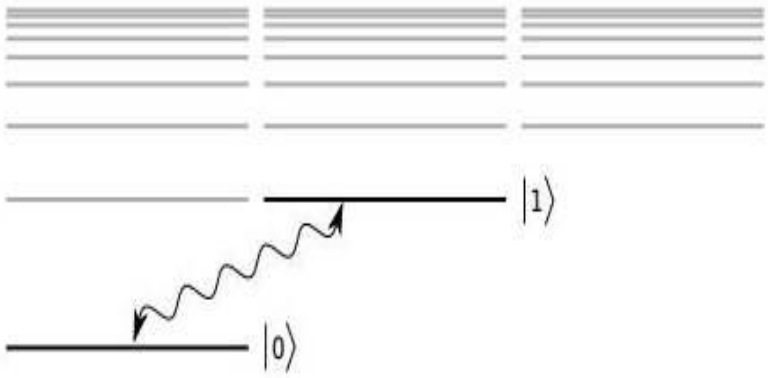


Рис. 1.4 – Диаграмма энергетических уровней атома. Основное состояние и первое возбужденное состояние соответствуют уровням кубита |0> и |1>

Типы квантового компьютера:

I.Кремниевый квантовый компьютер. Использование спина электрона в качестве квантового бита или кубита.

Рис. 1.5 – Кремниевые квантовые компьютеры

II. Оптический квантовый компьютер. Он использует фотон света в качестве кубита.

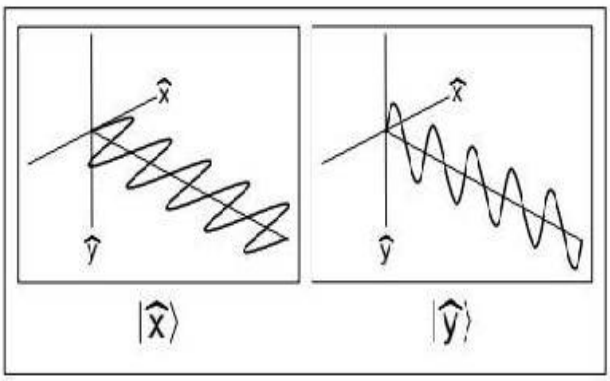


Рис. 1.6. – Горизонтальная поляризация соответствует состоянию кубита |x>, а вертикальное состояние соответствует состоянию кубита |y>

Поляризацию фотона можно измерить с помощью поляроидной пленки или кристалла кальцита. Подходящим образом ориентированный поляроидный лист пропускает фотоны с x-поляризацией и поглощает фотоны с y-поляризацией. Поэтому фотон, находящийся в суперпозиции

$|\varphi\rangle = a|0\rangle + b|1\rangle$ передается с вероятностью $|a|^2$. Если фотон теперь встречает другой лист поляроида с той же ориентацией, то он передается с вероятностью 1.

III. Secure Quantum Computer – Квантовый компьютер для безопасной связи.

Рис. 1.7 – Защищенный квантовый компьютер

IV. Квантовые вычисления модели шлюза — копируют цифровые вентили, которые являются строительными блоками современного компьютера, и создают квантовую эквивалентность для этих вентилей. Квантовые вычисления с крупнейшей моделью вентилей, которые позволяют сделать это сегодня, с учетом чисел 21, 7 и 3, поскольку мы все знаем, что это очень мелкомасштабный эксперимент.

Квантовые вычисления с ионной ловушкой: в качестве кубита используется ион. Вольфганг Пауль получил Нобелевскую премию по физике за работу в области квантовых вычислений на ионных ловушках. Ионы, или заряженные атомные частицы, можно удерживать и подвешивать в свободном пространстве с помощью электромагнитных полей. Кубиты хранятся в стабильных электронных состояниях каждого иона, а квантовая информация может обрабатываться и передаваться посредством коллективного квантованного движения ионов в ловушке (взаимодействующих посредством кулоновской силы). Лазеры применяются для создания связи между состояниями кубита (для операций с одним кубитом) или связи между внутренними состояниями кубита и внешними двигательными состояниями (для запутывания между кубитами).

V.D WAVE Quantum Computer: Совместный проект Google и НАСА.

Рис. 1.8– Квантовый компьютер D WAVE

11 мая 2011 года компания D-Wave Systems анонсировала D-Wave One, интегрированную квантовую компьютерную систему, работающую на 128-кубитном процессоре. Процессор, используемый в D-Wave One под кодовым названием «Rainier», выполняет одну математическую операцию — дискретную оптимизацию. Ренье использует квантовый отжиг для решения задач оптимизации. D-Wave One — первая в мире коммерчески доступная квантовая компьютерная система. Цена составит около 10 000 000 долларов США.

2. Цели исследования:

Основными задачами нашего исследования являются

I. Построить сверхбыстрые квантовые вычисления.

II. И сверхбезопасная квантовая связь.

III. Применение квантовых вычислений для бизнес-аналитики.

IV. Выяснить возможности квантовых вычислений в различных областях.

V. Разобраться в проблемах квантовых вычислений.

Потребность в квантовых вычислениях и их потенциальные преимущества перед обычными компьютерами изучались несколькими авторами (например, Канамори, Девитт, Прантош). Чен и Чаян (2012)

подробно обсудили процесс бизнес-аналитики и то, как анализ больших данных полезен для принятия бизнес-решений. Девитт и Манро (2011) реализовали модель высокопроизводительного квантового компьютера. Прантош (2011) обсудил будущее квантовой науки и ее масштабы в различных областях. Ааронсон обсудил в своем проекте ограничения квантовых вычислений при их практической реализации. Он подчеркивает, почему квантовому компьютеру для измерения импульса атома необходима абсолютная нулевая температура. В статье «Квантовый компьютер D WAVE» рассказывается о преимуществах и проблемах, связанных с квантовым компьютером D WAVE. В нем рассказывается, как можно использовать D WAVES при решении задачи коммивояжера и других сложных задач принятия решений [34-39].

В этом качественном исследовании мы пытаемся выяснить некоторые потенциальные преимущества и проблемы квантовых вычислений. Обычные компьютеры состоят из кремниевых чипов, на которых выгравированы более миллиардов транзисторов. Благодаря большему количеству транзисторов компьютеры становятся все быстрее и быстрее. Квантовая концепция кубита изменила всю концепцию вычислений. Обычный компьютер использует 8 бит только для хранения одного числа от 0 до 256, где, как и в квантовом компьютере, 8 кубитов могут одновременно хранить 256 чисел, что значительно ускоряет вычислительную мощность. Давайте рассмотрим все возможные комбинации 2-битной системы данных с 4 возможными состояниями 00, 01, 10 и 11. Классический 2-битный компьютер может одновременно выполнять максимум одну из этих 4 возможных функций. Чтобы проверить их все, компьютеру пришлось бы повторить каждую операцию отдельно, тогда как 2-битный квантовый компьютер благодаря явлению суперпозиции способен анализировать все эти возможности одновременно за одну операцию. Это связано с тем, что 2 кубита содержат информация о 4 состояниях, тогда как 2 бита содержат информацию об одном состоянии.

Таким образом, машина с n кубитами может одновременно находиться в суперпозиции 2^n состояний. Компьютер с 4 кубитами мог анализировать 16 параллельных состояний за одну операцию; для

сравнения, 4-битный классический компьютер может анализировать только одно состояние. Чтобы получить то же решение, что и квантовый компьютер, классический компьютер должен повторить эту операцию 16 раз.

10 кубитов — могут хранить 1024 числа.

11 кубитов – могут хранить 2048 чисел.

100 кубитов – можно хранить

1 267 650 600 228 229 401 496703205 376 чисел.

Мы можем сказать, что квантовый компьютер может решить проблему в масштабе, превосходящем любой обычный компьютер. Таким образом, мы обнаруживаем, что квантовые вычисления могут использоваться в огромных масштабах для анализа проблемы быстро растущих данных в сети, называемых большими данными. Его можно использовать для предварительного прогнозирования погоды, прогнозирования стихийных бедствий, таких как цунами, землетрясение, для бизнес-аналитики и многого другого. Врагом квантовых вычислений является среда, близкая к абсолютному нулю, очень чистая среда. Одной из самых больших проблем, с которыми сталкиваются ученые, работающие в области квантовых вычислений, является проблема декогерентности (изолировать систему от внешней среды), проблема оптимизации (выбора кратчайшего пути) и квантового туннелирования (вероятность исчезновения электрона на другой стороне).

В этой исследовательской работе показано несколько преимуществ квантовых вычислений и проблемы их реализации. Акцент сегодняшних вычислений делается на разработке и производстве такого компьютера, который обладает огромными вычислительными возможностями для анализа быстро растущих больших объемов данных в сети. Использование квантовых вычислений позволяет нам прогнозировать статистические выводы, принимать бизнес-решения, прогнозировать погоду, сопоставление шаблонов, анализ веб-данных и многое другое. Хотя квантовые вычисления находятся на начальной стадии, будущее вычислений и анализа больших данных больше зависит от квантовых компьютеров. Использование квантовых вычислений очень экологично по своей природе. Это может сэкономить

огромное количество тепла в центрах обработки данных и снизить энергопотребление с МВт до КВт. Существующие сегодня квантовые компьютеры, скажем, квант D WAVE НАСА, являются специфичными для конкретной области и используются для некоторых сложных приложений, а универсальный квантовый компьютер все еще остается для нас мечтой. Центр квантовых вычислений и коммуникационных технологий Австралии возглавляет глобальную гонку по разработке квантового компьютера и квантовой защищенной сети связи, а также имеет планы по разработке универсального квантового компьютера. Вполне возможно, что компьютер на 500 кубитов однажды сможет анализировать больше данных, чем атомов в наблюдаемой Вселенной.

1.2.Данные. Подходы и определения

Согласно ГОСТ Р 52653-2006[1], данные – представление информации в формализованном виде, пригодном для передачи, интерпретации и обработки.

Согласно ГОСТ 7.0-99[2], данные – информация, обработанная и представленная в формализованном виде для дальнейшей обработки.

В Кэмбриджском словаре приводятся следующие определения данных: данные – это информация, особенно факты и числа, собранные для последующего использования при принятии решений. Данные – это информация в электронной форме, пригодная для хранения и использования компьютером [13].

На сегодняшний день также довольно широко распространена формулировка, заключающаяся в том, что данные являются нефтью цифровой экономики [12].

Исходно понятие данных – философское, оно возникает в эпистемологии при рассмотрении основой проблемы гносеологии – познаваемости мира, поиска и осмысления истины. Процедуры верификации или фальсификации данных создают информацию, осмысление истины создает знание.

Философия рассматривает преобразование сведений в данные, данных в информацию, а информации – в знания. Истинность сведений

субъективна. Сведения, выраженные в формальном представлении, являются данными. Обработка данных позволяет определить сколько в них содержится информации. При осмыслении информации экспертом создаются знания.

Принятые в других странах термины могут отличаться, например, в США электронный документ обозначает любую информацию в цифровой форме, где “информация“ может включать в себя данные, текст, звуки, коды, компьютерные программы, программное обеспечение или базы данных. “Данные“ в этом контексте относятся к ограниченному набору элементов данных, каждый из которых состоит из содержимого или значения вместе с пониманием того, что означает контент или значение; где электронный документ содержит данные, это понимание того, что элемент данных или значение элемента данных должно быть явно включено в сам электронный документ или быть легко доступным для получателя электронного документа [4].

Жизненный цикл данных

Жизненный цикл данных – это последовательность этапов, которую конкретная порция данных проходит от начального этапа создания или получения до момента архивации или удаления [14].

Основные этапы жизненного цикла данных представлены на рис. 1.9.

Рассмотрим эти этапы подробнее.

Создание данных (Data Generation/Data Capture)

На этом этапе данные генерируются или захватываются. Этот этап обычно еще делят на три типа получения данных:

Приобретение данных (Data Acquisition)

Получение организацией данных, уже сгенерированных вне предприятия.

Запись данных (Data Entry)

Создание новых данных оператором или компьютером. Данные имеют ценность для предприятия.

Регистрация сигналов (Signal Reception)

Захват данных устройствами. Особенно важно в системах управления, но в последнее время особенно ценно при использовании такого подхода, как Интернет вещей.

Все три варианта получения данных очень важны в рамках рассмотрения процесса управления данными [17].

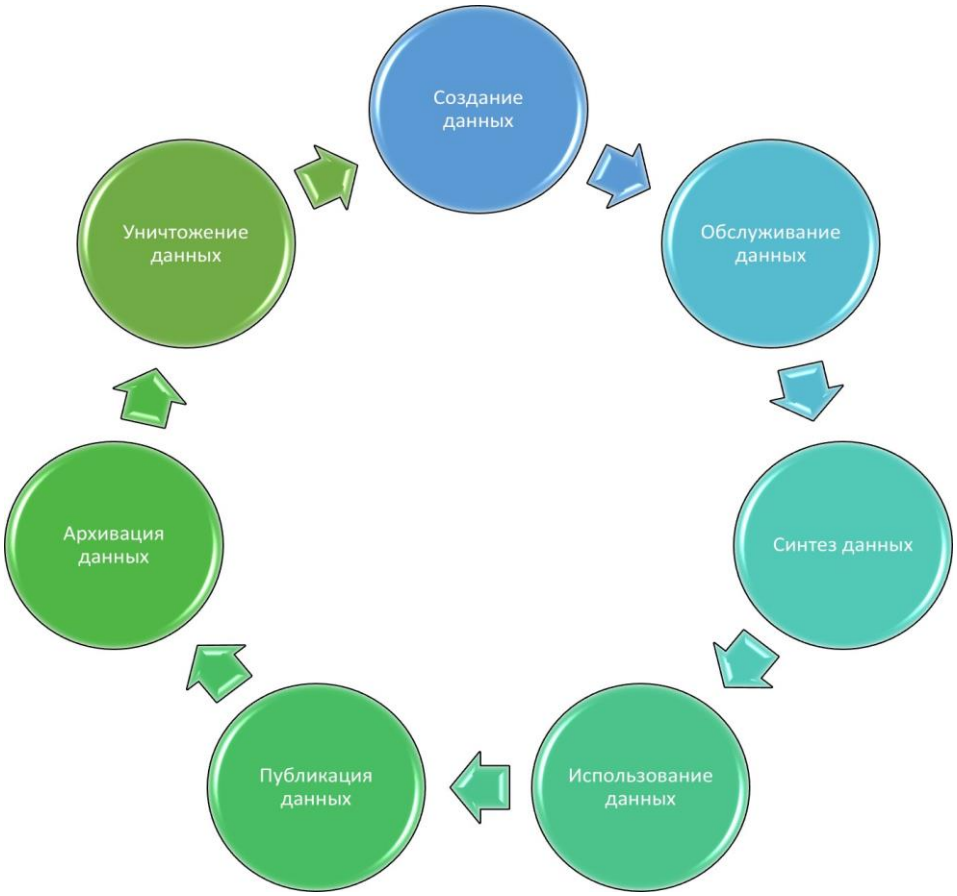


Рис. 1.9 – Жизненный цикл данных

Обслуживание данных (Data Maintenance)

После того, как данные были созданы, необходимо их хранить и обслуживать. Необходимо осуществлять доставку данных в точку, где их будут использовать или производить над ними манипуляции (например, операции синтеза).

Можно говорить о том, что обслуживание данных – это обработка данных без получения из извлечения из них полезной информации для предприятия.

Зачастую обслуживание данных включает в себя такие действия с данными, как перемещение, интеграция, очистка, обогащение и ETL-процессы (Extract, Transform, Load).

Обслуживание данных обычно подразумевает применение широкого спектра методов из области управления данными (Data Management) [17].

Синтез данных (Data Synthesis)

Это сравнительно недавно появившаяся стадия в жизненном цикле данных.

Используется не во всех моделях жизненных циклах данных.

Синтез данных – это процесс получения дополнительной ценности из данных при помощи использования индуктивной логики и сторонних информационных источников.

Это стадия, на которой с данными работаю аналитики, причем они могут использовать в своей работы методы моделирования рисков, актуарного моделирования, моделирования для принятия инвестиционных решений и др.

На этой стадии используется индуктивная логика, не дедуктивная. Индуктивная логика требует использования экспертного мнения, т.к. именно компетенции экспертов необходимы для построения моделей скоринга и др. [17].

Использование данных (Data Usage)

До сих пор шла речь об использовании данных внутри одного предприятия, которые возможно были подвержены очистке и обогащению на стадии обслуживания данных и использовались совместно с дополнительным третьими источниками данных на стадии синтеза данных.

На стадии использования данных они применяются в качестве полезной информации для задач, которые должны выполняться и управляться на основе данных.

Эти задачи могут быть вне жизненного цикла данных. Тем не менее, данные становятся все более значимой частью бизнес-процессов предприятий. Данные сами могут быть продуктом или услугой (или быть частью продукта или услуги), предлагаемой предприятием.

Использование данных имеет специальные задачи в рамках управления данными (Data Governance). Одна из задач заключается в законном использовании данных в требуемом виде. Это называется “разрешенное использование данных” (permitted use of data). Могут существовать регулирующие или договорные ограничения на то, как фактически можно использовать данные, а часть роли управления данными (Data Governance) заключается в обеспечении соблюдения этих ограничений [17].

Публикация данных (Data Publication)

При использовании данных возможна ситуация, когда данные отправляются за пределы предприятия. В этом случае говорят о публикации данных. Публикация данных – это вынос данных за пределы предприятия.

Примером этого процесса может быть маклер, рассылающий ежемесячные отчеты клиентам. Все данные, которые были разосланы, уже не могут быть отозваны. Если были разосланы данные с неверными значениям, то такие данные не могут быть исправлены, поскольку они уже становятся недоступны для предприятия. Управление данными (Data Governance) может потребоваться, чтобы помочь принять решение о том, как будут обрабатываться неверные данные, которые были отправлены из предприятия [17].

Архивация данных (Data Archival)

Данные могут быть использованы как однократно, так и несколько раз.

Но затем рано или поздно жизненный цикл данных начинает подходить к концу.

Первая стадия этого состояния заключается в архивации данных.

Архивация данных – это копирование данных в пассивную среду, в которой они хранится, для тех случаев, когда они понадобятся снова в активной производственной среде, и удаление этих данных из всех активных производственных сред.

Архив данных – это просто место, где хранятся данные, без их обслуживания, использования или публикации. В случае необходимости данные могут быть восстановлены из архива [17].

Уничтожение данных (Data Purging)

Уничтожение данных – это последовательность операций для выполнения необратимого удаления данных, делающая невозможным как восстановление данных, так и получение остаточной информации (Data Remanence) о них. Это одна из самых сложно реализуемых процедур управления данными.

Даже с теоретической точки зрения существует команда записи значения в ячейку памяти, но команды стирания значения как таковой нет. Для уничтожения данных необходимо изготовить высокопроизводительный источник случайных чисел и перезаписать ими весь носитель информации (перезаписи области хранения недостаточно, так как сохраняется информация об исходном количестве данных).

Иначе говоря, при уничтожении данных необходимо не только сделать недоступными от восстановления на физическом уровне сами данные, но и связанную с ними информацию в других наборах данных. Уничтожение данных регламентируется в ГОСТ Р 50735[3], причём классы защищенности данных.

и протоколы работы устанавливаются на уровне руководящих документов гостехкомиссии (ФСТЭК, Федеральной службы по техническому и экспортному контролю).

В настоящее время в РФ предусмотрено семь классов защиты информации. Согласно действующему на настоящему моменту пункту 19.2 приказа ФСТЭК N17 от 11.02.2013 “При выводе из эксплуатации машинных носителей информации, на которых осуществлялись хранение и обработка информации, осуществляется физическое уничтожение этих машинных носителей информации” [19].

Понятие метаданных

При сборе данных возникают метаданные, содержащие какую-либо информацию о собранных данных. Например, время создания набора данных, авторство и первоисточник, размер и кодировка данных – все это метаданные.

В соответствии с ГОСТ Р 52438-2005 метаданные – это сведения о данных.

Структурированные метаданные называют онтологией или схемой метаданных.

В методическом пособии “Онтологический инжиниринг знаний в системе PROTÉGÉ” [11] написано, что “онтология определяет общий словарь для ученых, которым нужно совместно использовать информацию в предметной области. Она включает машинно-интерпретируемые формулировки основных понятий предметной области и отношения между ними”.

Онтологии обычно используются в таких областях, как искусственный интеллект, семантическая паутина, системная инженерия, биомедицинская информатика, библиотечное дело, информационная архитектура и др. Все онтологии нужны для организации информации.

Жизненный цикл метаданных

Жизненный цикл метаданных можно разделить на четыре стадии, представленных на рис.1.10 [20] .

Оценка требований и анализ контента.

Спецификация системных требований.

Система метаданных.

Сервис и оценка.

Рассмотрим эти стадии подробнее.

Оценка требований и анализ контента

Этот этап состоит из четырех частей.

Выявление и получение базовых требований к метаданным.

Обзор релевантных стандартов и проектов по метаданным.

Исследование требований к глубоким метаданным.

Идентификация стратегий для схем метаданных.

Выявление и получение базовых требований к метаданным

На этом этапе жизненного цикла метаданных проводится опрос экспертов и специалистов по предметной области на тему требований к метаданным в конкретном проекте и анализ атрибутов. Целью опроса или интервьюирования экспертов является получение предварительной информации о проекте и установление контакта между специалистами по метаданным и провайдерами данных (или контента). На этой стадии должны быть уточнены и разъяснена и

уточнена область метаданных, контекст метаданных, действующая система метаданных, роль и функция метаданных [20] .

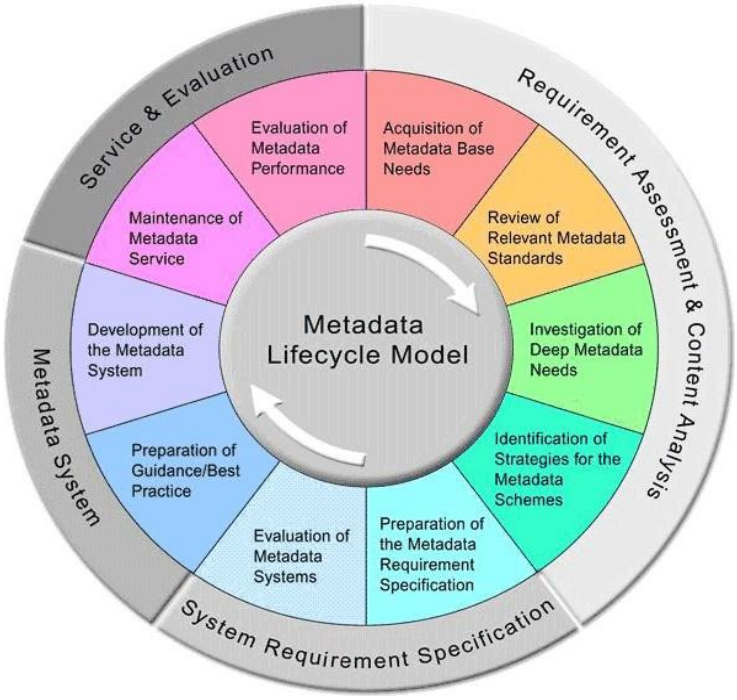


Рис. 1.10 – Жизненный цикл метаданных

Обзор релевантных стандартов и проектов по метаданным

Этот этап включает определение стандартов метаданных, потенциально полезных для проекта, изучение существующих схем метаданных и вариантов их использования. Этот этап нужен для того, чтобы можно было лучше узнать, какие различия существуют среди других аналогичных проектов, а также пересмотреть цели проекта [20].

Исследование требований к глубоким метаданным

На этом этапе определяются требования к метаданным более детально и глубоко. Для этого используется концепция контент-анализа. Здесь важно уточнить область и контент метаданных, а также проработан вопрос планируемых в работе СУБД и информационных систем, в которых планируется использовать метаданные. В дальнейшем эти наработки могут быть использованы в качестве методики для масштабирования [20] .

Идентификация стратегий для схем метаданных

На этом этапе формулируется стратегия использования метаданных на основе всех имеющих предыдущих результатов. Стратегия включает принятие одного или нескольких существующих стандартов метаданных и разработку на основе этих стандартов схемы метаданных [20] .

Спецификация системных требований

Этот этап состоит из двух частей.

Подготовка спецификации требований к метаданным

Оценка систем метаданных

Подготовка спецификации требований к метаданным

Спецификация требований к метаданным (Metadata Requirement Specification, MRS) является связующим звеном между участниками проекта, специалистами по метаданным и разработчиками систем. [20]

Оценка систем метаданных

На этом этапе производится оценка систем метаданных, которые потенциально могут использоваться в проекте в дальнейшем. Участники проекта могут выбрать из существующих систем метаданных. [20]

Система метаданных

Этот этап состоит из двух частей.

Подготовка руководства по лучшей практике.

Разработка системы метаданных.

Подготовка руководства по лучшей практике

Этот этап включает в себя создание документации и разработку руководящих принципов “лучшей практики” для отдельных элементов метаданных. Руководство может использоваться как контрольный список или как средство обеспечения контроля качества записей метаданных в проекте [20] .

Разработка системы метаданных

Разработчики системы разрабатывают инструменты и системы метаданных на основе спецификации требований к метаданным. [20]

Сервис и оценка

Этот этап состоит из двух частей.

Поддержка службы метаданных.

Оценка качества метаданных.

Поддержка службы метаданных

Цель разработки служб метаданных – гарантировать качество метаданных и механизмов работы с ними. Модель сервиса метаданных состоит из трех основных элементов: служебный механизм, роли и отношения между ролями. [20]

Оценка эффективности метаданных

Последний этап жизненного цикла метаданных направлен на рассмотрение результатов всего процесса работы с метаданными и качества самих метаданных. Интегральная оценка состоит из оценки качества метаданных, оценки эффективности работы схемы метаданных, оценки результатов использования инструментов создания метаданных и оценки применения модели жизненного цикла метаданных на каждом этапе [20].

1.3. Большие данные. Системы управления Большими данными

Если давать краткое определение, то *Большие данные* – это данные, которые не помещаются в оперативную память компьютера.

По сути это определение обозначает то, что свойство “быть большим” является не самостоятельным свойством данных, а зависит от характеристики системы, применяемой для их обработки.

Например, обычному человеку затруднительно запомнить какая именно температура была в нашем городе каждый день за прошедший месяц. Таким образом, три десятка значений вполне могут быть примером Больших данных. Однако вот человек уверенно сообщает “прошедший месяц был холодным”. Это сообщение несет информацию об обработанных данных: по мнению собеседника, средняя температура за прошедший месяц была ниже, чем обычно в этом месяце за несколько десятков лет.

Другим примером могут быть данные об объектах, которые теоретически несут важную информацию, однако имеющие такой размер, что эти данные практически невозможно не только обработать или сохранить, но даже собрать. Рассмотрим к примеру набор данных, содержащий координаты и скорости молекул в воздушном столбе над территорией аэропорта. Имеются также метаданные с описанием в

какой момент проводилось измерение и что это за молекула. Такой набор данных несет информацию о погодных условиях над аэропортом, включая температуру, давление, влажность, облачность, особые погодные условия – проходящий торнадо или падающий град. С другой стороны, для корректной обработки данные для всех молекул должны быть достаточно полны и репрезентативны для статистической обработки.

В результате такого мысленного эксперимента мы понимаем, что для эффективной работы с большими данными нужна модель данных, позволяющая сформировать методы работы с данными.

Данные могут быть различных типов. Информацию, полученную в результате учёта или измерения каких-либо объектов или параметров, называют *мастер-данными* (Master Data). Например, учёт количества, замеры координат и скоростей конкретных молекул – это мастер-данные.

Транзакционные данные (в англоязычной литературе применяются термины Transactional Data, Application Specific Data, Operational Data) – это данные, отображающие результат выполнения каких-либо операций. Например, данные о взаимодействии молекул между собой, а именно о пересечении границ рассматриваемой области, о траектории конкретной молекулы, об испарении капель дождя – это транзакционные данные. Транзакционные данные описывают взаимодействие объектов друг с другом или с окружающим миром, которые можно получить при помощи обработки мастер-данных.

Ретроспективные данные (Historical data) – это данные, снабженные метками времени. Например, с одной стороны мы можем сохранять данные о координате и векторе скорости каждой молекулы, но если у нас есть набор координат в зависимости от времени, то скорость молекулы становится лишней, она вычисляется исходя из модели, описываемой ньютоновской механикой.

Ссылочные данные (справочники, НСИ, нормативно-ссылочная информация, Reference Data, Lookup Data, Dictionaries) – это базовые неизменяемые данные, заранее известные из внешних источников, такие как нормативы, сокращения, акронимы, словари, стандарты. Например, удельные веса молекул, зависимость температуры замерзания и кипения

от давления, зависимость средней скорости молекул (скорости звука) от температуры.

Формат данных. Структурированные данные имеют заранее определенный формат. Полуструктурированные или слабоструктурированные данные – это данные, зачастую собранные из различных источников. Структура данных документирована, но в зависимости от источника данных конкретный формат представления информации может быть разным. Неструктурированные данные требуют обязательной обработки и последующей валидации перед использованием. Например, данные о координатах и скоростях молекул, в которых некоторые координаты пропущены или некоторые записи повторяются, являются полуструктурированными. Нам нужно понять, почему так произошло и перед использованием либо исключить такие данные (что может привести к систематической ошибке), либо, исходя из модели данных, восстановить пропущенные значения.

Данные, в которых координаты измеряются в разных единицах измерения, числа иногда записаны словами, иногда латинскими цифрами, а иногда в виде сканированного изображения почерка лаборанта, являются неструктурированными данными.

Обычно Большие данные описываются при помощи следующих характеристик [3].

Объем (Volume) – количество сгенерированных и хранящихся данных. Размер данных определяет значимость и потенциал данных, а также то, могут ли они быть рассмотрены как Большие данные.

Разнообразие (Variety) – тип данных. Большие данные могут состоять из текста, изображений, аудио, видео. Большие данные при сопоставлении друг с другом могут дополнять отсутствующие данные.

Скорость (Velocity) – скорость. Здесь подразумевается скорость, с которой данные генерируются и обрабатываются. Очень часто Большие данные используются в режиме реального времени.

Изменчивость (Variability) – противоречивость наборов данных может препятствовать их обработке и управлению ими.

Достоверность (Veracity) – качество данных напрямую влияет на точность проведения анализа данных.

На рис. 1.11 при помощи интеллект-карты показаны компоненты экосистемы Больших данных. Рассмотрим эти компоненты подробнее.

Распределенные файловые системы

Для хранения и обработки Больших данных созданы распределенные системы хранения данных, в том числе *распределенные файловые системы*, позволяющие использовать внешнее файловое пространство системы хранения для обработки данных на нодах, входящих в вычислительных кластер.

Зачастую удобно использовать распределенные файловые системы, арендуемые как отдельный облачный сервис, например, Google Colossus[1], Amazon S3[2], Yandex Disk[3].

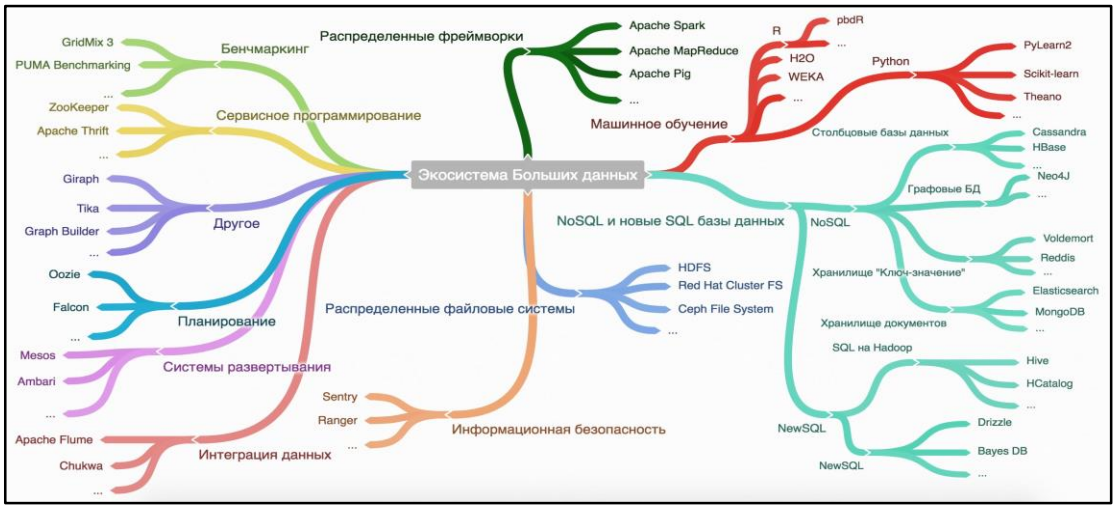


Рис. 1.11 – Интеллект-карта экосистемы Больших данных

Обработка находящихся на распределенных системах хранения данных ведется параллельно на компьютерах, составляющих узлы (nodes) *вычислительного кластера*.

---

1 Google Colossus. https://cloud.google.com/bigtable/docs/overview

2 Amazon S3. https://aws.amazon.com/ru/s3/

3 Yandex Disk. https://tech.yandex.com/disk/

Планирование

Инструменты *планирования* позволяют автоматизировать повторяющиеся задачи и запускать задания на основе таких событий, как добавление нового файла в папку. Они похожи на такие инструменты, как CRON в Linux, но специально разработаны для работы в отказоустойчивом кластере. Вы можете использовать их, например, для запуска задачи MapReduce всякий раз, когда в каталоге имеется новый набор данных [2].

Системы развертывания

Настройка инфраструктуры Больших данных – непростая задача, и развертывание новых приложений в кластере Больших данных – это зона ответственности инженеров по Большим данным. Они в значительной степени автоматизируют установку и настройку компонентов Больших данных [2].

Интеграция данных

Допустим, что уже есть распределенная файловая система, и теперь необходимо перенести данные из одного источника в другой. В таких случаях используют *фреймворки для интеграции данных*, такие как Apache Sqoop и Apache Flume. Этот процесс похож на процесс извлечения, преобразования и загрузки (Extract, Transform and Load, ETL) в традиционном хранилище данных [2].

Информационная безопасность

Средства обеспечения безопасности Больших данных позволяют осуществлять централизованный контроль доступа к данным. Безопасность Больших данных стала самостоятельной дисциплиной, и дата-ученые обычно сталкиваются с ней только как потребители данных. Безопасностью Больших данных занимаются эксперты по информационной безопасности [2].

Машинное обучение

Если у вас есть Большие данные, то было бы неплохо получить из них полезный контент. Это можно сделать при помощи использования методов машинного обучения, статистики и прикладной математики [2].

На сегодняшний день, когда появилось огромнейшее количество данных, один компьютер уже не в состоянии справиться с задачей их обработки. Некоторые алгоритмы, разработанные в прошлом веке, увы,

не смогут справиться с этой задачей, даже если теоретически можно было бы подключить к решению задачи все компьютеры Земли. Это связано с временной сложностью алгоритма[4].

С примерами временной сложности можно ознакомиться на Stack Overflow[5].

Одна из самых больших проблем со старыми алгоритмами заключается в том, что они недостаточно масштабируются. Учитывая объем данных, которые необходимо анализировать сегодня, это становится проблематичным. Для обработки этого объема данных требуются специализированные структуры и библиотеки. Например, в языке Python есть следующие библиотеки: Scikit-learn (библиотека машинного обучения), PyBrain (для работы с нейронными сетями), NLTK (для обработки естественного языка), Pylearn2 (еще одна библиотека машинного обучения), TensorFlow (библиотека глубокого обучения, есть программный интерфейс API для языка Python), Keras (библиотека для работы с нейронными сетями) и другие [2] .

Существует также Apache Spark – программный каркас с открытым исходным кодом для реализации распределенной обработки неструктурированных и слабоструктурированных данных[6].

Базы данных NoSQL и новые SQL базы данных

Использование реляционных баз данных для обработки Больших данных крайне неэффективно из-за высоких накладных расходов. Традиционно для обработки Больших данных используются базы данных типа “ключ – значение” (Key value database или HashDB). Одна из первых баз этого типа DBM была реализована Ken Thompson для AT&T Unix 7 в 1979 году.

---

[4] Временная сложность алгоритма. https://ru.wikipedia.org/wiki/Временная_сложность_алгоритма

[5] Real-world example of exponential time complexity. http://stackoverflow.com/questions/7055652/real-world-example-of-exponential-time- complexity

[6] Apache Spark. http://spark.apache.org/

База данных вида “ключ – значение”, по сути, представляет собой ассоциативный массив (Hash, Dict), то есть множество, состоящее из пар (Key, Value). В некоторых реализациях на множестве ключей вводится отношение порядка, и мы можем получить значения последовательно по мере возрастания ключа. В других случаях сортировка по ключу неустойчива при одинаковых ключах и при неоднократных выборках можно получить различную последовательность пар.

Большое количество баз данных можно разделить на следующие типы:

Столбцовые базы данных (Column databases). Данные хранятся в столбцах, что позволяет алгоритмам выполнять гораздо более быстрые запросы.

Хранилища документов (Document stores) Хранилища документов больше не используют таблицы, но сохраняют каждое наблюдение в документе. Это позволяет использовать гораздо более гибкую схему данных.

Потоковые данные (Streaming data). Данные собираются, преобразуются и агрегируются не в партиях, а в реальном времени.

Хранилища для ключей (Key-value stores). Данные не хранятся в таблице; для каждого значения назначается ключ (как рассказано об этом выше).

SQL на Hadoop – пакетные запросы на Hadoop, использующие фреймворк

MapReduce в фоновом режиме.

Новый SQL (New SQL). Этот тип сочетает масштабируемость баз данных NoSQL с преимуществами реляционных баз данных. Здесь используется интерфейс SQL и реляционная модель данных.

Графовые базы данных (Graph databases). Это тип баз данных, использующих графовые структуры для семантических запросов с узлами и ребрами и свойствами для представления и хранения данных. Классическим примером этого типа является социальная сеть.

Глава 2. КВАНТОВЫЕ ВЫЧИСЛЕНИЯ

2.1. Квантовые состояния и кубиты

Прежде всего, для понимания модели квантовых вычислений необходимо овладеть понятийным аппаратом, который используется для описания и работы. Вся терминология пришла прямиком из квантовой механики, поэтому для тех, кто вполне владеет квантово-механическим аппаратом, понимание модели квантовых вычислений будет простым (хотя и здесь придётся приложить усилия к изучению нескольких терминов, пришедших из теории информации и теории вычислений). Здесь будут даны самые простейшие определения терминов. Тем же чита телям, кто захочет полностью и глубоко понять математический аппарат, лежащий за описываемой моделью, имеет смысл обратиться к серьёзной литературе по квантовой механике.

Перед рассмотрением основного понятия в модели квантовых вычислений — кубита, необходимо изучить понятие квантового состояния. Квантовым состоянием будем называть совокупность из некоторого символа (наименования квантового состояния) и приписанного к нему коэффициента, причём этот коэффициент является комплексным числом. Квантовое состояние будет записываться как [34]:

$$\alpha|s\rangle,$$

где α — комплексночисленный коэффициент, а s — наименование квантового состояния. Последнее обычно состоит из одного символа, например: «0», «1», «+», «–». Так что, квантовыми состояниями, например, являются такие объекты, как $|0\rangle$ и $|1\rangle$. А можно придумать и более сложные квантовые состояния, например — $^{1-i}|\mathrm{t}\rangle$.

Кубитом, в этом случае, называется просто список квантовых состояний. Это слишком общее определение, и в других книгах по квантовым вычислениям обычно даётся иное определение. Однако здесь мы заострим внимание на этом аспекте — кубит состоит из списка квантовых состояний, при этом есть одно ограничение — сумма квадратов модулей всех комплексночисленных коэффициентов обязательно должна равняться 1.

Обычно и всегда это введённое таким образом понятие ограничивают. Поскольку традиционный бит представляет собой возможность выбора из двух альтернатив (0 или 1), то и кубит

ограничивают двумя квантовыми состояниями в списке. Далее станет понятно, почему именно так, а теперь имеет смысл перейти к рассмотрению понятия «базис».

Базисом называется набор кубитов, которые взаимно ортогональны друг другу. Если ограничивать рассмотрение кубитов двумя квантовыми состояниями, то, само собой разумеется, базис состоит из двух кубитов. Однако же что такое «ортогональность» в применении к кубитам? Если кубит — это всего лишь список комплексных чисел, к которым приписаны некоторые наименования квантовых состояний, что как могут быть ортогональны такие «именованные комплексные числа»?

Всё просто. Дело в том, что базис выбирается тоже произвольным образом. Для простоты понимания и соответствия традиционному пониманию бита базисом назван набор кубитов

$|0\rangle$ и $|1\rangle$, после чего было введено векторное представление кубита. Для этих двух базисных кубитов векторное представление следующее:

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix},$$
$$|1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}.$$

И, собственно, ортогональность в таком случае определяется как равенство нулю скалярного произведения двух векторных представлений кубитов. Но что из себя представляют эти числа 0 и 1 в векторах? Это ничто иное, как комплексночисленные коэффициенты α при базисных кубитах, то есть так и получается, что $|0\rangle = 1|0\rangle + 0|1\rangle$, $|1\rangle = 0|0\rangle + 1|1\rangle$. Таким образом, произвольный кубит можно разложить в базисе, и такое разложение записывается как $|\varphi\rangle = \alpha|0\rangle + \beta|1\rangle$, и оно называется *линейной суперпозицией базисных состояний,* и, соответственно, в виде вектора такой кубит представляется как:

$$\begin{pmatrix} \alpha \\ \beta \end{pmatrix},$$

где α и β — некоторые комплексные числа такие, что сумма квадратов их модулей равна строго 1.

Всё дело в том, что комплексный коэффициент при базисном кубите, или, что то же, при базисном квантовом состоянии, — это так называемая *амплитуда вероятности*. Это понятие из

квантовой механики, и оно обозначает тот простой факт, что при измерении вероятность обнаружения кубита в этом квантовом состоянии равна квадрату модуля его амплитуды. Именно поэтому сумма квадратов модулей должна равняться строго единице, поскольку при измерении кубит будет обнаружен либо в том, либо в другом базисном квантовом состоянии [35].

Но что такое измерение? Это понятие пришло непосредственно из квантовой механики, где оно определяется достаточно сложно. Здесь же для упрощения рассмотрим такое определение. Измерение кубита — это попытка ответить на вопрос типа «Находится ли данный кубит в квантовом состоянии |0⟩?» или «Каковы амплитуды вероятности нахождения данного кубита в квантовых состояниях, определяемых некоторым базисом?». Ответом на первый вопрос становится амплитуда вероятности нахождения кубита в запрашиваемом квантовом состоянии, а ответом на второй вопрос — набор амплитуд вероятностей, соответствующих базисным квантовым состояниям, сумма которых, конечно же, равна единице.

Необходимо отметить, что, как следует из положений квантовой механики, после измерения кубит переходит в какоелибо состояние из числа входящих в базис, в рамках которого производится измерение, при этом вероятность перехода кубита в это состояние равна как раз квадрату модуля амплитуды при этом состоянии в базисе. Понять это проще всего на нескольких незамысловатых примерах, однако перед их рассмотрением стоит более подробно изучить используемую нотацию.

Читатель уже заметил эти странные скобки — |s⟩. Это так называемая «нотация Дирака». Придумана она давнымдавно, однако именно такой её внешний вид позволяет с лёгкостью понимать смысл вычислений и применять модель квантовых вычислений. Прежде всего, надо отметить, что |s⟩ называется «кет-вектором», а ⟨*s*|, соответственно, «бравектором». Словечки «бра» и «кет» — это две части английского слова «bracket», которое переводится как «скобка». Собственно, кет-вектор — это вектор столбец, которые мы уже видели при обсуждении векторного представления кубитов. А бра-вектор — это комплексно-сопряжённый с соответствующим кет-вектором вектор-строка. То есть, например, если у нас есть некоторый кубит

$$|\varphi\rangle = \begin{pmatrix} \alpha \\ \beta \end{pmatrix},$$

то соответствующим бра-вектором будет

$$\langle\varphi|=\left(\alpha^*\beta^*\right),$$

где α^* и β^* — комплексные сопряжённые чисел α и β соответственно (для комплексного числа $a + bi$ комплексным сопряжённым будет комплексное число $a - bi$).

Из этого следует простое мнемоническое правило — если соединить браи кет-вектор, чтобы получился «бракет», то это будет *скалярным произведением* двух векторов [36]:

$$\langle\varphi|\varphi\rangle=\left(\alpha^*\beta^*\right)\begin{pmatrix}\alpha\\ \beta\end{pmatrix}=\alpha^*\alpha+\beta^*\beta=|\alpha|^2+|\beta|^2,$$

если комплексное число умножить на его комплексное сопряжённое, то, без всяких сомнений, получится квадрат модуля этого числа, что каждый читатель может легко проверить при помощи формулы перемножения многочленов).

А что будет, если соединить браи кет-векторы в обратном порядке, то есть в виде «кетбра»? Это, само собой разумеется, записывается как $|\varphi\rangle\langle\varphi|$, а поскольку это векторы, то результатом такого их перемножения явится матрица. Если векторы имеют размерность 2, то такая матрица будет иметь размерность 2x2:

Это так называемая *матрица плотности* $\rho=|\varphi\rangle\langle\varphi|$. Этот термин ещё пригодится при детальном изучении модели квантовых вычислений.

Таким образом, есть два простых мнемонических правила:

1. $\langle\varphi|\varphi\rangle$ — скалярное произведение, которое называется «бракет» (от англ. *bracket*).
2. $|\varphi\rangle\langle\varphi|$ — матрица плотности, что проще всего запомнить в виде эдакого неформального преобразования $|\varphi\rangle\langle\varphi|\to\varphi\times\varphi$, то есть обращённые одна к другой угловые скобки как бы превратились в знак умножения.

$$|\varphi\rangle\langle\varphi|=\begin{pmatrix}\alpha\\ \beta\end{pmatrix}\left(\alpha^*\beta^*\right)=\begin{pmatrix}\alpha\alpha^* & \alpha\beta^*\\ \beta\alpha^* & \beta\beta^*\end{pmatrix}.$$

Теперь можно более чётко понять, что такое измерение. Например, чтобы ответить на вопрос «Находится ли данный кубит $|\varphi\rangle$ в квантовом состоянии $|0\rangle$?», необходимо просто выполнить операцию $\langle 0|\varphi\rangle$, и ответом на вопрос станет квадрат модуля полученного в результате вычисления числа. Само собой разумеется, что для квантового состояния $|0\rangle$ это тривиально, но то же самое

правило действует и для других квантовых состояний — необходимо просто посчитать скалярное произведение, взять модуль результата и возвести его в квадрат.

Всё это понять ещё легче, если обратиться к графическому представлению. Несмотря на то, что до этого момента для иллюстрации понятий квантовое состояние и кубит использовались некоторые символы и манипуляции ими, этим символам можно дать определённые интерпретации, в частности — графическую. Но для этого для начала необходимо несколько упростить задачу — пусть коэффициенты перед квантовыми состояниями будут не комплексными, а действительными. В этом случае всё очень просто: каждый кубит представляет собой единичный вектор, а его разложение в базисе — это всего лишь проекции данного вектора на произвольно выбранные ортогональные «оси» [37]:

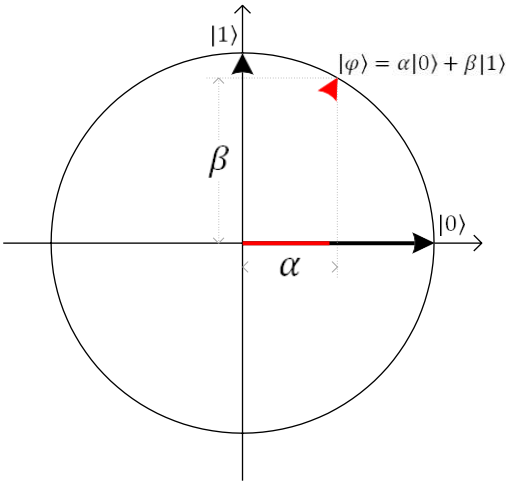


Рис. 2.1. Графическое представление разложения кубита в базисе в случае действительных коэффициентов

В этом случае становится понятным, почему $\alpha^2+\beta^2=1$ — это всего лишь теорема Пифагора (конечно же, окружность в данном случае всегда имеет радиус 1).

Тут так же необходимо отметить, что «оси», то есть базисные квантовые состояния выбраны абсолютно произвольно — единственное условие, которому они должны удовлетворять, — это ортогональность. Просто так уж сложилось по традиции, что выбирают горизонтальный вектор $|0\rangle$ и вертикальный вектор $|1\rangle$. Но, само собой разумеется, это не единственный базис. Базисов может

существовать бесконечное количество — любая пара ортогональных единичных векторов может служить базисом.

Например, другим часто используемым базисом является базис {|+⟩, |−⟩}:

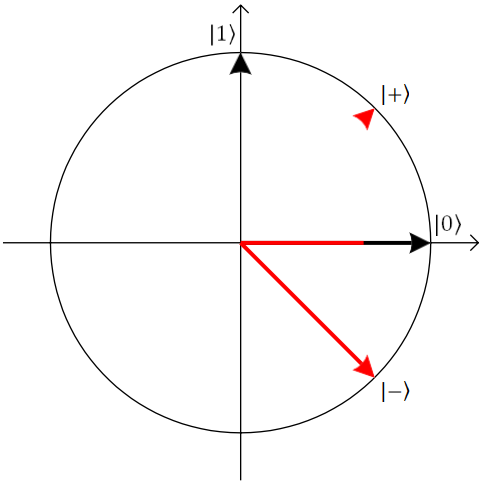


Рис.2.2. Базис {|+⟩, |−⟩} и его расположение относительно стандартного базиса

Естественно, что оба квантовых состояния этого базиса можно выразить через основной базис {|0⟩, |1⟩}, и сделать это довольно просто. Даже не вдаваясь в серьёзные расчёты (хотя и это можно было бы сделать), но используя только упомянутую ранее теорему Пифагора, можно вычислить значения коэффициентов α и β, которые в данном случае равны. Это делается так (помним, что α в данном случае — действительное число, поэтому квадрат его модуля равен просто квадрату этого числа):

$$2\alpha^2 = 1 \Rightarrow \alpha^2 \frac{1}{2} \Rightarrow \alpha = \frac{1}{\sqrt{2}},$$

то есть:

$$|+\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle,$$

$$|-\rangle = \frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle.$$

Этот базис также часто используется в модели квантовых вычислений, поэтому его хорошо бы постоянно держать в уме. По крайней мере, формулы пересчёта из одного базиса в другой полезно помнить наизусть.

Эти формулы помогут, к примеру, отвечать на такие вопросы, как «Если выполнить измерение заданного кубита в базисе {|+⟩, |−⟩}, то с какой вероятностью кубит примет ние |+⟩?» и подобные. Ведь если есть кубит, для которого имеется

выражение в стандартном базисе $|\varphi\rangle$, то для ответа на этот вопрос достаточно произвести операцию $\langle +|\varphi\rangle$, и в результате будет получена амплитуда искомой вероятности.

Например, если кубит $|\varphi\rangle$ разлагается в стандартном базисе как $|\varphi\rangle = \alpha|0\rangle + \beta|1\rangle$, то для перевода его в базис $\{|+\rangle, |-\rangle\}$ можно воспользоваться следующей формулой:

$$|\varphi\rangle_{+|-} = \langle +|\varphi\rangle|+\rangle + \langle -|\varphi\rangle|-\rangle = \frac{\alpha+\beta}{\sqrt{2}}|+\rangle + \frac{\alpha-\beta}{\sqrt{2}}|-\rangle .$$

Эта формула поясняется при помощи диаграммы на следующем рисунке:

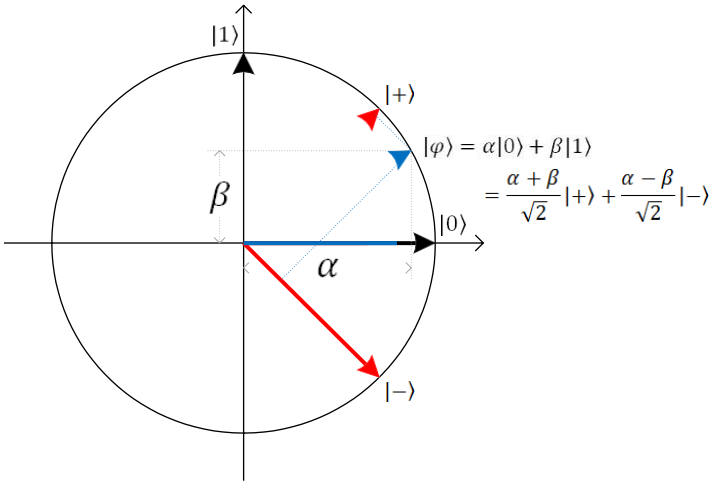


Рис. 2.3. Перевод кубита в базис $\{|+\rangle, |-\rangle\}$

Собственно, матричные операции дают возможность не только ответить на первый вопрос об измерении, но и на второй, то есть сразу узнать каковы амплитуды вероятности нахождения данного кубита в квантовых состояниях, определяемых некоторым базисом? Для этого надо перемножать не векторы, а сразу умножить сопряжённую матрицу, представляющую собой базис, на векторное представление кубита. В результате получится вектор, представляющий собой именно амплитуды вероятностей.

Например, матрица стандартного базиса выглядит как [38]

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

и её комплексно-сопряжённая матрица выглядит точно так же. Если эту матрицу умножить на представление кубита в стандартном базисе, то в результате, как ни странно, получатся коэффициенты α и β, то есть амплитуды вероятности нахождения кубита в базисных состояниях $|0\rangle$ и $|1\rangle$. А вот матрица базиса $\{|+\rangle, |-\rangle\}$ выглядит так:

$$\begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix},$$

и её комплексно-сопряжённая матрица абсолютно такая же. Соответственно, можно произвести умножение комплексносопряжённой матрицы на кубит $|\varphi\rangle = \alpha|0\rangle + \beta|1\rangle$:

$$\begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} \frac{\alpha+\beta}{\sqrt{2}} \\ \frac{\alpha-\beta}{\sqrt{2}} \end{pmatrix},$$

что полностью соответствует полученному ранее результату.

Все эти примеры, которые рассматривались выше, имеют один нюанс — коэффициенты в квантовых состояниях кубитов в них действительные. Честно говоря, это нисколько не уменьшает силу квантовой вычислительной модели, однако в объективном мире уж так устроено, что квантовая механика оперирует с комплексными коэффициентами. Такова модель описания реальности, принятая в квантовой механике, а потому она перенесена и в вычислительную модель, поскольку в конце концов кубиты будут реализованы в «железе», и такая реализация кубитов будет иметь дело именно с комплексными коэффициентами. Поэтому здесь надо рассмотреть дополнительную визуализацию модели при помощи диаграммы, но уже с квантовыми коэффициентами.

Если действительные коэффициенты при двух базисных квантовых состояниях приводили к появлению одномерного пространства состояний (окружность), то резонно предположить, что использование комплексных коэффициентов приведёт к появлению двумерного пространства — сферы. Так оно и есть, и такая визуализация называется *сферой Блоха*.

Тут есть один тонкий момент (те, кто не любит формул, могут сразу же перейти к следующей странице и посмотреть на приятную взору диаграмму). Поскольку каждый кубит нормирован, его длина всегда равна единице (то есть, напомним, $|\alpha|^2 + |\beta|^2 = 1$), то разложение в стандартном базисе можно записать как

$$|\varphi\rangle = e^{i\gamma}\left(\cos\frac{\theta}{2}|0\rangle + e^{i\omega}\sin\frac{\theta}{2}|1\rangle\right).$$

Это выражение для кубита $|\varphi\rangle$ осуществляется на основании широко известной формулы Эйлера ($e^{ix} = \cos x + i \sin x$). Поскольку

коэффициенты α и β являются комплексными числами, то ничто не мешает записать их как систему уравнений:

$$\alpha = e^{i\gamma}\cos\frac{\theta}{2} = \cos\gamma\cos\frac{\theta}{2} + i\sin\gamma\cos\frac{\theta}{2}$$

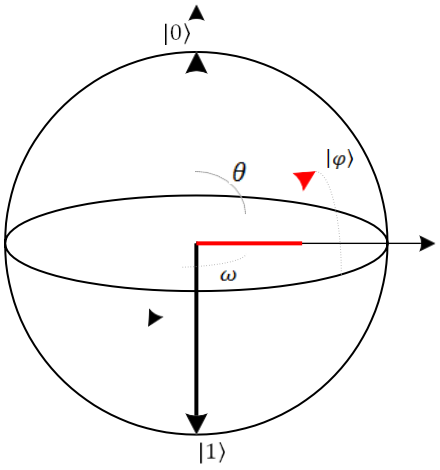
$$\beta = e^{i\gamma}e^{i\omega}\sin\frac{\theta}{2} = e^{i(\gamma+\omega)}\sin\frac{\theta}{2} = \cos(\gamma+\omega)\sin\frac{\theta}{2} + i\sin(\gamma+\omega)\sin\frac{\theta}{2}.$$

Так уж получилось, что в квантовом мире множитель $e^{i\gamma}$ можно опустить, поскольку он не приводит к каким-либо наблюдаемым эффектам — это просто некоторый фазовый коэффициент, который при проведении измерения никак не влияет на результаты. Это значит, что в данном случае приведённые выше уравнения можно переписать в виде:

$$\alpha = \cos\frac{\theta}{2} + i\sin\frac{\theta}{2},$$

$$\beta = \cos\omega\sin\frac{\theta}{2} + i\sin\omega\sin\frac{\theta}{2}.$$

Так что кубит описывается двумя угловыми параметрами — θ и ω, а они приводят к чёткой геометрической интерпретации кубита с комплексными коэффициентами в виде точки на сфере [39]:



Как видно, в данном представлении базис представляется парой квантовых состояний, не ортогональных друг другу, но разнонаправленных и лежащих на одной прямой. На диаграмме показан стандартный базис, но вдумчивый читатель может решить вышеприведённые уравнения для базиса {|+⟩, |−⟩} и отобразить его на сфере Блоха.

И теперь осталось упомянуть об одной важной особенности операции измерения в модели квантовых вычислений, которая, если можно так выразиться, контр-интуитивна. Измерение — это дорога в один конец. Другими словами, если мы произвели измерение кубита и получили какой-то конкретный результат (с определённой вероятностью), то сам кубит принял значение, полученное в

результате измерения, и вся суперпозиция базисных состояний была потеряна. Восстановить её невозможно.

Таким образом, после измерения вся квантовая информация, которая хранилась посредством суперпозиции базисных состояний с комплексными коэффициентами, теряется, а остаётся только одна из двух альтернатив. Именно поэтому кубит и называется «битом» — несмотря на то, что потенциально в нём в «скрытом виде» хранится бесконечное количество информации (произвольная суперпозиция двух ортонормированных базисных состояний), при измерении из него можно выудить только один бит информации — либо одно, либо другое состояние в базисе измерения.

Несколько кубитов

Итак, из рассмотренного до сего времени сложно сделать выводы о том, что модель квантовых вычислений даёт какое-то преимущество по сравнению с традиционной вычислительной моделью. Ну да, один кубит содержит в скрытом виде бесконечное количество информации, но мы никак не можем этим воспользоваться. Это именно скрытая информация, которая используется внутри кубита неизвестным для нас способом. В чём же дело?

Интересная особенность квантовых вычислений проявляется тогда, когда на сцену выходят многокубитовые системы. Дело в том, что правила комбинации кубитов разительно отличаются от правил комбинации битов. Проще всего рассмотреть вариант комбинации двух кубитов.

В квантовой механике и при обычном рассмотрении модели квантовых вычислений считается, что любые два кубита всегда находятся в разных пространствах состояний, поэтому даже если обозначения их базисов одинаковые, то всё равно считается, что базисы разные (и для этого иногда используют индексы,

которые тут же начинают опускать). Однако это — усложнение рассмотрения, поэтому далее будет считаться, что все кубиты находятся в одном и том же пространстве состояний и раскладываются на суперпозицию в одинаковых базисах.

Так что пусть есть два кубита $|\varphi\rangle$ и $|\psi\rangle$. Они представляют собой две суперпозиции базисных состояний, скажем, в стандартном базисе: $|\varphi\rangle = \alpha_1|0\rangle + \beta_1|1\rangle$ и $|\psi\rangle = \alpha_2|0\rangle + \beta_2|1\rangle$ — это просто два обычных кубита. Но что будет, если их соединить в двухкубитовую систему? Всё просто:

$$|\varphi\rangle|\psi\rangle = \alpha_1\alpha_2|0\rangle|0\rangle + \alpha_1\beta_2|0\rangle|1\rangle + \beta_1\alpha_2|1\rangle|0\rangle + \beta_1\beta_2|1\rangle|1\rangle$$

Для упрощения обозначений запись $|\varphi\rangle|\psi\rangle$ сокращают до $|\varphi\psi\rangle$, и это так называемое *тензорное произведение* кубитов или, ещё шире, тензорное произведение квантовых систем. Внезапно система из двух кубитов представляет собой суперпозицию четырёх базисных состояний (а то, что эти состояния — $|00\rangle$, $|01\rangle$, $|10\rangle$ и $|11\rangle$ являются базисными, мы скоро убедимся). Вдумчивый читатель уже понял, что добавление третьего кубита в квантовую систему приводит к появлению суперпозиции восьми состояний. Так что система из n кубитов представляет собой суперпозицию из 2^n квантовых состояний. Не это ли экспоненциальное увеличение мощности вычислительной модели?

Квантовая модель вычислений говорит, что вычисления с кубитом производятся одновременно со всеми базисными состояниями, на которые кубит разложен. Поскольку для одного кубита имеет место два базисных состояния, при работе с одним кубитом вычисления одновременно производятся с двумя квантовыми состояниями. Таким образом, при работе с многокубитовой системой вычисления одновременно производятся с экспоненциальным числом квантовых состояний, как показано в предыдущем абзаце.

Учёные говорят, что во всей Вселенной всего имеется около 10^{80} всевозможных атомов, а это число примерно равно 2^{266}, то есть если сделать немыслимый традиционный компьютер из всей Вселенной, в котором каждый атом будет представлять один бит информации, то этот компьютер сможет успешно смоделировать только 266 кубита. Ну вот как-то так можно представить себе вычислительную мощность модели квантовых вычислений.

Таким образом, многокубитовая система представляет собой не просто набор кубитов, а нечто большее, поскольку кубиты взаимосвязаны друг с другом, и суперпозиция образуется на всю систему, а не на каждый кубит в отдельности. Это значит, что для многокубитовой системы при разложении на суперпозицию базисных состояний должен определяться новый базис. Предыдущее рассмотрение уже показало, что в таком базисе должно быть 2^n базисных состояний для n-кубитовой системы.

Что такое тензорное произведение векторов? Проще всего понять суть выражения $|00\rangle$ следующим образом. Необходимо

вспомнить векторное представления квантового состояния $|0\rangle$, после чего произвести операцию тензорного произведения:

$$|00\rangle = |0\rangle|0\rangle = \begin{pmatrix}1\\0\end{pmatrix}\begin{pmatrix}1\\0\end{pmatrix} = \begin{pmatrix}1 & \begin{pmatrix}1\\0\end{pmatrix}\\ 0 & \begin{pmatrix}1\\0\end{pmatrix}\end{pmatrix} = \begin{pmatrix}1\\0\\0\\0\end{pmatrix},$$

Точно таким же образом выходит следующее представление других квантовых состояний двухкубитовой системы:

$$|01\rangle = \begin{pmatrix}0\\1\\0\\0\end{pmatrix},$$

$$|10\rangle = \begin{pmatrix}0\\0\\1\\0\end{pmatrix},$$

$$|11\rangle = \begin{pmatrix}0\\0\\0\\1\end{pmatrix}.$$

Можно простейшими вычислениями убедиться, что эти четыре вектора образуют ортонормированный базис в четырёхмерном комплекснозначном пространстве. И, соответственно, матрица этого базиса выглядит следующим образом:

$$\begin{pmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{pmatrix}.$$

Несложные умозаключения позволяют понять, что для системы из n-кубитов стандартным базисом является единичная матрица размера $2^n \times 2^n$. Но, само собой разумеется, что базисов может быть бесконечное число: единственное ограничение — это ортонормированность всех векторов.

А что, если в формуле разложения в стандартном базисе для двух кубитов, выполнить некий финт ушами? Ведь ничто не запрещает нам сделать вот какую-то такую суперпозицию квантовых состояний:

$$|\varphi\rangle = \frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle.$$

Эта суперпозиция отвечает главному свойству — её коэффициенты, то есть амплитуды вероятности, соответствуют условию равенства единице суммы квадратов модулей. Так что такое разложение системы из двух кубитов вполне может существовать. Однако как оно получено? Ведь действительно, можно легко показать, что не существует таких двух кубитов, чьё тензорное произведение давало бы такой результат. Достаточно рассмотреть систему уравнений [40]:

$$\begin{cases} \alpha_1\alpha_2 = \dfrac{1}{\sqrt{2}} \\ \alpha_1\beta_2 = 0 \\ \beta_1\alpha_2 = 0 \\ \beta_1\beta_2 = \dfrac{1}{\sqrt{2}} \end{cases}$$

чтобы понять, что у неё нет решения. Так что такая квантовая суперпозиция была получена как-то иначе (её реально можно получить на физических объектах — элементарных частицах, так что это не игра разума теоретиков). Действительно, это так называемое *запутанное состояние*, и у таких состояний есть множество применений в рамках модели квантовых вычислений. В частности, квантовая телепортация, о которой можно прочитать в большинстве источников по квантовым вычислениям (поэтому мы не будем её здесь описывать), работает именно с запутанными состояниями.

В практических целях выделяют четыре подобных состояния, которые называются состояниями Белла по имени одного из исследователей парадокса Эйнштейна — Подольского — Розена, в котором формулируются вопросы, на которые отсутствуют ответы в рамках классической физики. Джон Стюарт Белл придумал чрезвычайно хитроумный способ доказать при помощи эксперимента, что в квантовой механике нет так называемых «скрытых параметров», и что сама природа мира оперирует неопределённостью, которая «схлопывается» в момент измерения. Но эта занимательная история выходит за рамки данной книги, поэтому заинтересованного читателя можно отослать к более детальной литературе по квантовой механике.

Итак, есть четыре состояния Белла, которые, кстати, образуют ортонормированный базис (что легко показывается — скалярное произведение любой пары этих состояний равно 0). Вот они:

$$\left|\Phi^{+}\right\rangle = \frac{1}{\sqrt{2}}\left|00\right\rangle + \frac{1}{\sqrt{2}}\left|11\right\rangle,$$
$$\left|\Phi^{-}\right\rangle = \frac{1}{\sqrt{2}}\left|00\right\rangle - \frac{1}{\sqrt{2}}\left|11\right\rangle,$$
$$\left|\Psi^{+}\right\rangle = \frac{1}{\sqrt{2}}\left|01\right\rangle + \frac{1}{\sqrt{2}}\left|10\right\rangle.$$
$$\left|\Psi^{-}\right\rangle = \frac{1}{\sqrt{2}}\left|01\right\rangle - \frac{1}{\sqrt{2}}\left|10\right\rangle.$$

В общем, эти состояния рекомендуется выучить и запомнить — они понадобятся во многих приложениях модели квантовых вычислений. Очень многие квантовые алгоритмы используют их. Более того, именно наличие таких запутанных состояний позволяет осуществлять некоторые «фишки» квантовых вычислений, недоступные в классической вычислительной модели, поскольку в ней попросту нет аналога этому явлению.

Вот появились некие свойства модели квантовых вычислений, которые, вроде бы, повышают эффективность этой модели по сравнению с классическими вычислениями. Однако здесь не всё так просто. Дело в том, что сама теория сложности вычислений является областью настолько неразведанной, что сказать абсолютно точно об эффективности квантовых вычислений по сравнению с вычислениями обычными ничего нельзя. Да, разработаны квантовые алгоритмы, которые для некоторых задач показывают кардинальное увеличение эффективности. Например, для задачи разложения числа на простые множители пока не существует эффективного алгоритма в классической модели вычислений, а в модели квантовых вычислений такой алгоритм есть. Но никто пока не смог доказать, что такой же эффективный алгоритм в конце концов не будет найден и для классических вычислений. Так что вопрос о том, является ли модель квантовых вычислений эффективнее традиционной модели, остаётся открытым.

Область эта — всё ещё terra incognitæ, и нас ждёт множество открытий в теории сложности вычислений. Но пока мы даже не знаем, равны ли классы сложности **P** (алгоритмы, работающие за полиномиальное время) и **NP** (алгоритмы, результат работы которых можно проверить за полиномиальное время). Если окажется так, что равны, то вряд ли можно сказать, что модель квантовых вычислений эффективнее классической вычислительной модели.

2.2. Гейты и квантовые схемы

Теперь мы готовы перейти собственно к вычислениям. Всё рассмотренное перед этим разделом касалось базиса и кубитов (ну и многокубитовых состояний), однако о вычислениях пока речи не шло. Сейчас же переходим к вычислениям и определяем, как и какие операции можно производить над кубитами.

Читатель, знакомый со схемной нотацией вычислительных процессов над классическими битами, должен вспомнить, что вычисление в традиционной вычислительной модели может быть определено при помощи нескольких равнозначных формализмов — машины Тьюринга (в самых разных её вариантах), лямбда-исчисления и, наконец, при помощи булевых схем. Этот способ мы сейчас и вспомним.

Вычислительный процесс представляет собой функцию, которая в общем случае определяется следующим образом [34-35]:

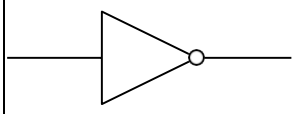
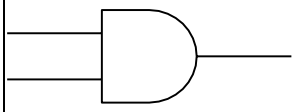
$$f : \{0,1\}^n \to \{0,1\}^m$$

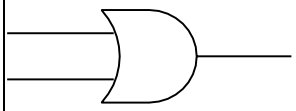
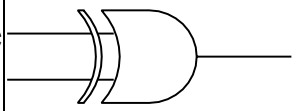
то есть функция является отображением упорядоченного набора из n битов в упорядоченный набор из m битов.

Собственное, такие функции можно «конструировать» из мелких блоков, которые традиционны для схемотехники. Вот, к примеру, некоторые из традиционно использующихся блоков:

Таблица 2.1.

Основные схемотехнические элементы

Функция	**Схемотехническое обозначение**	**Таблица истинности**
НЕ		f: {0,1} → {0,1} f 0 = 1 f 1 = 0
И		f: $\{0,1\}^2$ → {0,1} f 0 0 = 0 f 0 1 = 0 f 1 0 = 0 f 1 1 = 1

ИЛИ		f: $\{0,1\}^2 \rightarrow \{0,1\}$ $f\ 0\ 0 = 0$ $f\ 0\ 1 = 1$ $f\ 1\ 0 = 1$ $f\ 1\ 1 = 1$
Исключающее ИЛИ		f: $\{0,1\}^2 \rightarrow \{0,1\}$ $f\ 0\ 0 = 0$ $f\ 0\ 1 = 1$ $f\ 1\ 0 = 1$ $f\ 1\ 1 = 0$

Как видно, это обычные логические операции — отрицание, конъюнкция, дизъюнкция, исключающее ИЛИ и т. д. Они используются в совершенно разных областях деятельности, но были введены именно в рамках традиционной вычислительной модели как полноценная замена формализмам машины Тьюринга и лямбда-исчисления.

Эти функции или схемотехнические элементы также могут составлять базис в том смысле, что имеются некоторые наборы (в том числе, минимальные по количеству элементов), через элементы которых можно выразить произвольную функцию любой размерности. Приведённый в таблице 1 набор элементов является базисным (но не минимальным — в нём есть избыточные элементы, которые могут быть выражены через другие элементы; например, элемент ИЛИ может быть выражен при помощи комбинации двух элементов НЕ и одного элемента НЕ-И посредством известного правила де Моргана). При помощи него, например, можно составить следующую небезынтересную функцию:

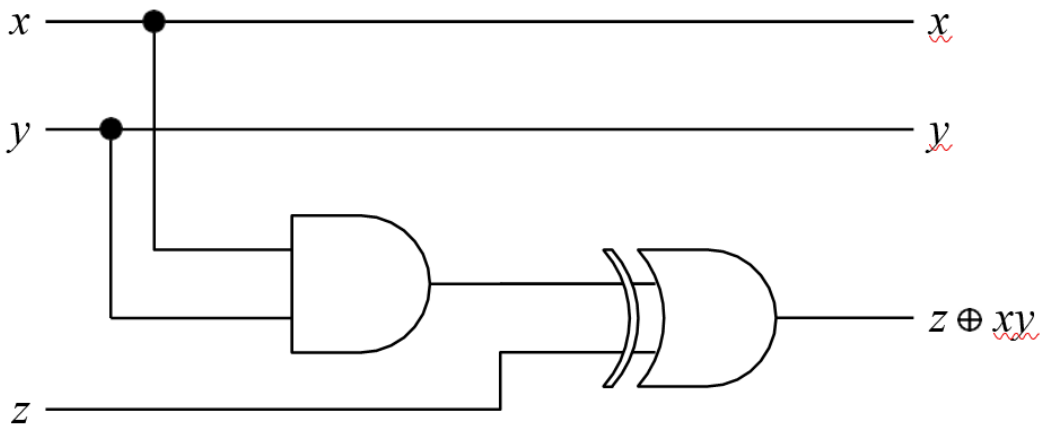


Рис. 2.4. Схемное представление элемента Тоффоли

Это так называемый *элемент Тоффоли* (или, как ещё называют, «вентиль Тоффоли»), замечательная функция, которая одна сама по себе образует минимальный базис — через неё одну можно выразить любую другую функцию общего вида. Её таблица истинности выглядит следующим образом:

Таблица 2.2.

Результат работы элемента Тоффоли

Вход			**Выход**		
x	y	z	x	y	$z \oplus xy$
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	1	0
0	1	1	0	1	1
1	0	0	1	0	0
1	0	1	1	0	1
1	1	0	1	1	1
1	1	1	1	1	0

Действительно, если мы определим эту функцию следующим образом (и здесь мы начинаем использовать для пояснения язык программирования Haskell):

```
toffoli :: Int -> Int -> Int -> (Int, Int, Int)toffoli 1 1 0 = (1, 1, 1)
toffoli 1 1 1 = (1, 1, 0)toffoli x y z = (x, y, z)

getX, getY, getX :: (Int, Int, Int) -> IntgetX (x, _, _) = x
```

```
getY (_, y, _) = y
getZ (_, _, z) = z
```

тогда все другие базовые функции (из таблицы 1) можно определить в точности через неё и только через неё одну (с учётом вспомогательных функций «добывания» компонентов тройки):

```
not :: Int -> Int
not = getZ . toffoli 1 1

and :: Int -> Int -> Int
and a b = getZ $ toffoli a b 0

nand :: Int -> Int -> Int
nand a b = getZ $ toffoli a b 1

or :: Int -> Int -> Int
or a b = not $ and (not a) (not b)

nor :: Int -> Int -> Intnor a b = not $ or a b

xor :: Int -> Int -> Int
xor a b = getZ $ toffoli 1 a b
```

Несколько некорректными выглядят определения функций or и nor, поскольку они выражены не напрямую через функцию toffoli, а при помощи использования правил булевой логики. Тем не менее, такой подход тоже возможен. Ну а вдумчивому читателю предлагается поразмыслить и попробовать выразить эти функции через функцию toffoli при помощи фиксации тех или иных аргументов, либо при помощи каких-либо иных способов [36].

При этом надо отметить, что сама по себе функция not является обратимой (то есть взаимно однозначной, по значению её выхода можно доподлинно восстановить значение входа), поэтому её выражение через элемент Тоффоли вообще является очень неэффективным и неэкономным. Это значит, что при разработке алгоритмов для реализации функций в квантовой вычислительной модели разработчик всегда должен обращать внимание на свойства функции. Есть такие функции, которые обратимы сами по себе (как функция not или дискретное преобразование Фурье, о котором много будет рассказано далее), и их можно реализовать вообще без потерь энергии.

Кроме этого интересного элемента есть ещё один широко описываемый в литературе — это так называемый *элемент Фредкина*. Его отличие от только что рассмотренного элемента заключается в том, что из трёх его входных битов на выходе меняются два, а не один, как у элемента Тоффоли. Вот соответствующая таблица истинности [35]:

Таблица 2.3.

Результат работы элемента Фредкина

Вход			**Выход**		
C	I_1	I_2	C	O_1	O_2
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	1	0
0	1	1	0	1	1
1	0	0	1	0	0
1	0	1	1	1	0
1	1	0	1	0	1
1	1	1	1	1	1

То есть видно, что изменения происходят на предпоследней и третьей с конца строках. На самом деле, изменения производятся тогда, когда бит C равен 1, и изменения эти заключаются в том, что значения двух других битов меняются местами. То есть, если C = 1, то $O_1 = I_2$, а $O_2 = I_1$. Когда $I_1 = I_2$ изменений, конечно же, не видно.

Этот элемент обладает теми же свойствами, что и предыдущий рассмотренный — он один образует базис функций, так как при помощи комбинаций из одного лишь этого элемента можно построить любую другую функцию. Для читателя будет интересным попробовать сделать это, а также выразить элемент Тоффоли через элемент Фредкина и наоборот.

Если внимательно посмотреть на элементы Тоффоли и Фредкина, то будет видна одна особенность. На поверхности видно то, что оба этих элемента имеют одинаковое количество входных и выходных битов. Но если взглянуть поглубже, то окажется, что это неспроста. По результату, то есть по выходным битам элемента можно однозначно восстановить все входные биты. Этого

нельзя сказать ни об одном элементе, который приведён ранее в таблице, кроме элемента НЕ.

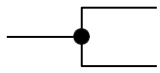
Это свойство называется *обратимостью*, и это очень важное свойство. В 1961 году Рольф Ландауэр сформулировал принцип и вывел формулу, оценивающую нижнюю оценку количества выделяемой теплоты (то есть рассеяния энергии в окружающее пространство) при потере одного бита информации. Это был важный шаг, который позволил связать воедино два вида энтропии — термодинамическую и информационную. И этот принцип сегодня подтверждается необходимостью иметь огромные охлаждающие мощности для серверных в центрах обработки данных — классические вычисления постоянно теряют информацию бит за битом мириады раз в секунду, что приводит к выделению большого количества теплоты.

Здесь также будет уместным упомянуть, что эти два рассмотренных элемента также могут быть использованы для реализации так называемого элемента FANOUT, который дублирует поданный на его вход бит. В классической логике наличие этого элемента подразумевается как бы

«по умолчанию» (на рис. 5 элемент FANOUT обозначен жирной точкой на линии), однако в обратимых вычислениях эта операция не может быть осуществлена просто так, поскольку если взять её именно такой, какой она подразумевается в классической модели, она будет необратимой. Естественно, что выражение при помощи элементов Тоффоли и Фредкина использует один из выходных битов для того, чтобы дублировать значение одного из входных битов. Например, если на первый вход элемента Тоффоли подать 1, а на третий — 0, то на третьем выходе будет дубликат второго входного бита (который также будет присутствовать на втором выход). Примерно так же реализуется элемент FANOUT при помощи элемента Фредкина (читателю, как всегда для тренировки ума предлагается подумать, как именно; но в любом случае любознательный читатель сможет найти пример реализации в прилагаемом к книге файле).

Таблица 2.4.

Элемент FANOUT

Функция	**Схемотехническое обозначение**	**Таблица истинности**
FANOUT		$f: \{0,1\} \rightarrow \{0,1\}^2$ $f\ 0 = (0, 0)$ $f\ 1 = (1, 1)$

Модель квантовых вычислений основана на квантовой механике, одним из положений которой является обратимость её законов во времени. Любая эволюция квантовой системы обратима во времени, и поэтому данное свойство должно быть непременно отражено в модели квантовых вычислений. А это означает, что любой вычислительный процесс в рамках квантовых вычислений должен быть обратимым.

Так что если в природе имеется какой-либо обратимый процесс, который неким способом производит аналоговые вычисления элемента Тоффоли или элемента Фредкина, то при помощи этого процесса можно построить универсальное вычислительное устройство, которое является обратимым. Собственно, в рамках квантовой механики такие процессы уже найдены. Но перейдём к математическому описанию, поскольку вопросы физической реализации модели квантовых вычислений в этой книге не рассматриваются.

Теперь посмотрим, что это означает с точки зрения математики (а мы в самом начале договорились, что математику мы привлекаем в качестве метанауки, позволяющей манипулировать символами без необходимости вникания в смысл этих символов). В модели квантовых вычислений каждое вычисление осуществляется при помощи так называемых *гейтов*. Гейт в терминах квантовой механики — это «унитарное преобразование», то есть некоторая квадратная матрица U, размерность которой соответствует количеству базовых квантовых состояний, подчиняющаяся простому свойству:

$$U^{\dagger}U = UU^{\dagger} = I,$$

где $U^{\dagger}$ есть эрмитово-сопряжённая матрица к U. Эрмитовое сопряжение является операцией над квадратными матрицами с комплексными числами. Для того чтобы получить

эрмитовосопряжённую матрицу, необходимо транспонировать матрицу и заменить все числа в ней на их комплексно-сопряжённые. Например:

$$\begin{pmatrix} 1 & -3i \\ 2+i & 3 \end{pmatrix}^{\dagger} = \begin{pmatrix} 1 & 2-i \\ 3i & 3 \end{pmatrix}.$$

Ну, в общем-то, представленный пример показывает просто эрмитово-сопряжённую матрицу, но она не является унитарным преобразованием, которое может использоваться в рамках модели квантовых вычислений, поскольку если перемножить эти две матрицы, то в результате точно не получится единичная матрица *I* (кто из читателей хочет, может проверить и сравнить свой результат с тем, что приведён на врезке на следующей странице).

Понятно, что такие преобразования являются обратимыми в модели квантовых вычислений, поскольку если мы применим к какому-либо кубиту некоторое унитарное преобразование, то для получения первоначального состояния кубита мы можем применить эрмитовое сопряжение этого унитарного преобразования, в результате чего будет получен первоначальный кубит:

$$U^{\dagger}(U|\varphi\rangle) = U^{\dagger}U|\varphi\rangle = (U^{\dagger}U)|\varphi\rangle = I|\varphi\rangle = |\varphi\rangle.$$

Можно рассмотреть несколько важных примеров унитарных преобразований (или, как их ещё называют, *унитарных операторов* в гильбертовом пространстве), действующих на одном кубите. К примеру, самым простым, тривиальным оператором является единичная матрица, поскольку эрмитово-сопряжённой к ней является она сама же, а произведение двух единичных матриц есть единичная матрица.

А вот ещё несколько замечательных матриц, у которых даже есть собственные названия, поскольку они очень часто используются в квантовых алгоритмах. Это так называемые *матрицы Паули*, роль которых в модели квантовых вычислений заключается в поворотах кубитов вокруг различных осей в двумерном пространстве. Вот эти матрицы (единичная матрица *I* также часто включается в этот набор в качестве матрицы с индексом 0):

$$\sigma_1 = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \sigma_2 = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, \sigma_3 = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}.$$

В рамках модели квантовых вычислений для этих матриц используются более подходящие обозначения: *X*, *Y* и *Z*. Можно посмотреть, что получается, если эти операторы применять к

кубитам в стандартном вычислительном базисе. Например, изучим действие матрицы *X*:

$$X|0\rangle = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}\begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix} = |1\rangle,\ X|1\rangle = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}\begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} = |0\rangle,$$

$$X|\varphi\rangle = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}\begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} \beta \\ \alpha \end{pmatrix}.$$

Этот унитарный оператор можно считать квантовым аналогом логического элемента НЕ, поскольку он переводит кубит $|0\rangle$ в $|1\rangle$ и наоборот. У суперпозиции базовых вычислительных состояний этот оператор просто меняет местами коэффициенты при базисных состояниях. В общем-то, это можно считать квантовой операцией отрицания.

Теперь рассмотрим действие оператора *Y*. Примерно то же самое [36]:

$$Y|0\rangle = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}\begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ i \end{pmatrix},\ Y|1\rangle = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}\begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} -i \\ 0 \end{pmatrix},$$

$$Y|\varphi\rangle = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}\begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} -\beta i \\ \alpha i \end{pmatrix}.$$

Это так называемый оператор сдвига фазы, и у него нет аналогов в традиционной логике. А оператор *Z* просто меняет знак у коэффициента при базовом состоянии $|1\rangle$ — для того чтобы понять это, не надо даже перемножать матрицы, достаточно просто взглянуть на матрицу этого оператора.

Четыре матрицы Паули (*I*, *X*, *Y* и *Z*) образуют полный базис унитарных операторов в двумерном гильбертовом пространстве. У них есть ещё множество интересных и важных свойств, которые более подробно можно изучить в специализированной литературе. Ну а читателю в качестве упражнения для разминки мозгов рекомендуется умозрительно понять, какие преобразования осуществляют эти операторы над точкой (кубитом) на сфере Блоха.

Теперь можно рассмотреть гейт Адамара, поскольку он очень примечателен тем, что переводит кубиты из стандартного вычислительного базиса в базис $\{|+\rangle, |-\rangle\}$ и обратно. Этот гейт обозначается как H и имеет следующую матрицу:

$$H = \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix}.$$

Легко понять, что кубит $|0\rangle$ переводится в $|+\rangle$, а кубит $|1\rangle$ — в $|-\rangle$, и обратно. Этот унитарный оператор часто используется в квантовых алгоритмах, поэтому читателю рекомендуется хорошенько его запомнить, а также для тренировки преобразовать несколько разных кубитов при помощи этого гейта.

Гейты бывают не только однокубитовые (все рассмотренные ранее гейты являются именно однокубитовыми). Само собой разумеется, что унитарное преобразование может воздействовать на систему кубитов произвольного размера. В этом случае размерность матрицы унитарного преобразования равно размерности базиса, то есть 2^k для k кубитов. Вот, к примеру, наиболее важный двухкубитовый гейт, который называется *CNOT* («контролируемое НЕ»):

$$CNOT = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}.$$

Действие этого гейта понять просто — он применяет ко второму кубиту гейт X тогда и только тогда, когда первый кубит равен $|1\rangle$. То есть имеет место система равенств:

$$CNOT|00\rangle = |00\rangle,$$
$$CNOT|01\rangle = |01\rangle,$$
$$CNOT|10\rangle = |11\rangle,$$
$$CNOT|11\rangle = |10\rangle.$$

Этот гейт осуществляет, по сути, квантовую условную операцию — условием является значение первого кубита, и если оно равно $|1\rangle$, ко второму кубиту применяется гейт X (квантовое НЕ). Вообще говоря, любое унитарное преобразование U можно сделать контролируемым CU. Для этого достаточно добавить ещё один кубит, от значения которого зависит, применяется ли преобразование U или нет. Если значение контрольного кубита равно $|0\rangle$, то преобразование U не применяется, и все кубиты проходят через него без изменений. В противном случае преобразование U применяется, и результатом является применение этого преобразования к своему множеству кубитов. В виде формул этого записывается следующим образом ($\otimes$ — операция тензорного произведения):

$$CU(|0\rangle \otimes |\varphi\rangle) = |0\rangle \otimes |\varphi\rangle,$$

$$CU(|1\rangle \otimes |\varphi\rangle) = |1\rangle \otimes (U|\varphi\rangle).$$

Матрица такого контролируемого унитарного преобразования U выглядит следующим образом (обобщённый вид):

$$CU = \begin{pmatrix} I & 0 \\ 0 & U \end{pmatrix},$$

где I — единичная матрица $2^n \times 2^n$, где n — количество кубитов в регистре $|\varphi\rangle$.

Есть ещё множество интересных гейтов как для одного кубита, так и для многих. Все их описания могут быть найдены в соответствующей литературе, а в этой книге в случае необходимости специальные гейты будут описываться непосредственно в рамках описания квантовых алгоритмов, в которых используются. Теперь же имеет смысл рассмотреть то, как гейты обозначаются. Всё просто — большинство гейтов обозначаются прямоугольником с входами и выходами, причём число выходов в точности равно числу входов (вычисления обратимы). В самом прямоугольнике указывается наименование гейта:

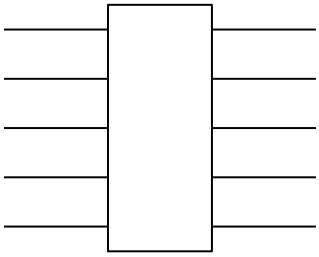


Рис. 2.5. Условное обозначение произвольного гейта U на квантовых схемах

Если необходимо указать, какие кубиты подаются на тот или иной вход, то эти кубиты указываются непосредственно около входа. Обычно для запуска процесса квантовых вычислений все входные кубиты инициализируются в значение $|0\rangle$, и если около входа не указан кубит (как на рисунке вверху), то считается, что на этот вход подаётся именно значение $|0\rangle$.

Собственно, взглянув на рис. 6 и становится понятным, почему именно функциональное программирование стоит ближе всего к модели квантовых вычислений, чем любая другая парадигма программирования. Гейт U на рис. 6 имеет вполне определённый смысл в рамках функционального программирования — это функция, которая возвращает ровно столько же данных, сколько принимает на

вход. То есть гейт — это некий достаточно ограниченный вид функций.

Особое обозначение имеется у гейта *CNOT*, как у «квантового условного оператора». Он обозначается так, как указано на следующем рисунке:

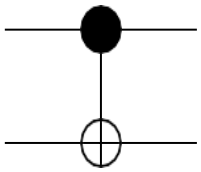


Рис. 2.6. Условное обозначение гейта *CNOT*

Соответственно, чёрный кружок обозначает контролирую щий кубит,то естьв зависимости от значения которого к контролируемому кубиту применяется квантовая операция отрицания. Соответственно, контролируемый кубит обозначается кружком с крестом, и это легко запомнить, поскольку в логике символом $\oplus$ обозначают логическую операцию «Исключающее ИЛИ» (XOR), а именно так можно определить результат контролируемого кубита: $|b'\rangle = |a\rangle \oplus |b\rangle$, где a — контролирующий кубит, а b — контролируемый. При этом необязательно, что контролирующий кубит находится сверху, на некоторых схемах его удобнее рисовать снизу.

Контролируемый гейт *CU* обозначается схожим образом:

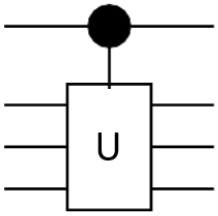


Рис. 2.8. Условное обозначение гейта *CU*

Часто в квантовых алгоритмах используются так называемые *оракулы*, которые являются квантовыми аналогами чёрных ящиков. Если есть некоторая функция на двоичных строках f: $\{0,1\}^n \rightarrow \{0,1\}^m$, то соответствующим оракулом для неё будет являться некоторое унитарное преобразование U_f: $\{0,1\}^{n+m} \rightarrow \{0,1\}^{n+m}$, которое считается заданным и вычисляющим эту функцию f. Считается как бы, что есть некоторый физический процесс, который обратимым образом вычисляет эту функцию, осуществляя квантово-механическое унитарное преобразование. Это кажется

несколько странным, но для некоторых классов функций такие физические процессы найдены. Обозначаться такой оракул будет следующим образом:

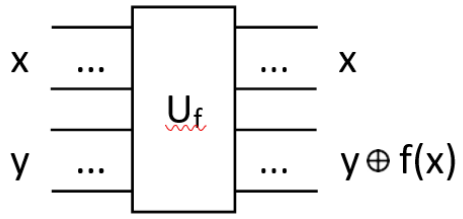


Рис. 2.9. Условное обозначение оракула U_f

Здесь регистр x является входными данными для функции f и этот параметр занимает n кубитов. Регистр y является вспомогательным (и обычно, но не обязательно, проинициализированным в строку нулей) и занимает m кубитов. На выходе результат вычисления функции складывается по модулю 2 с этим регистром.

С другой стороны квантовая модель вычислений обладает не меньшей вычислительной силой по сравнению с классическими вычислениями. Это значит, что любая функция, которая может быть вычислена при помощи классической схемы, также может быть вычислена и при помощи последовательности унитарных преобразований. Имеется квантовый аналог элемента Тоффоли, и, соответственно, любая бинарная функция может быть построена при помощи преобразования её логической схемы к представлению в виде элементов Тоффоли (ниже будет приведён пример подобного представления). Таким образом, в общем случае для построения квантового оракула для произвольной функции $f:\{0,1\}^n \to \{0,1\}^m$ можно применить метод её конструирования из базисных элементов. В итоге получается обратимое унитарное преобразование $U_f:\{0,1\}^{n+m} \to \{0,1\}^{n+m}$, которое можно использовать в алгоритме.

Наконец, процесс (или операция) измерения кубита обозначается стилизованным изображением измерительного прибора:

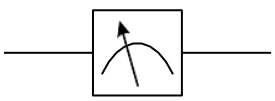


Рис. 2.10. Условное обозначение операции измерения

Само собой разумеется, что эти обозначения гейтов являются как бы строительными блоками при рисовании так называемых

квантовых схем, то есть диаграмм для визуализации квантовых алгоритмов. Вот пример одной квантовой схемы:

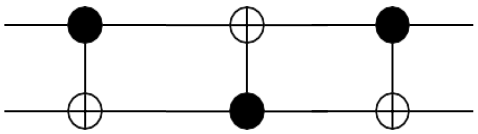


Рис. 2.10. Пример квантовой схемы

Эта квантовая схема примечательна тем, что меняет местами квантовые состояния двух кубитов, поданных на её вход. Это можно легко проверить для базисных состояний двухкубитовой квантовой системы {|00⟩, |01⟩, |10⟩, |11⟩} при помощи следующей таблицы:

Таблица 2.5.

Результат работы квантовой схемы с рис. 2.10

Входные кубиты	**Результат первого гейта**	**Результат второго гейта**	**Результат третьего гейта**	
|00⟩	|00⟩	|00⟩	|00⟩	
|01⟩	|01⟩	|11⟩	|10⟩	
|10⟩	|11⟩	|01⟩	|01⟩	
|11⟩	|10⟩	|10⟩	|11⟩	

То есть матричное представление этой квантовой схемы выглядит так (надеюсь, читатель к этому моменту в книге уже вполне может оперировать матричными представлениями ку-битов и многокубитовых систем):

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

Этот же результат можно получить, если перемножить три матрицы тех унитарных операторов, которые представлены на этой квантовой схеме. Необходимо при этом помнить, что квантовая схема читается слева направо, то есть самый левый гейт применяется к входным кубитам первым, а значит перемножать соответствующие матрицы необходимо как бы в обратном порядке (если три гейта на рис. 11 обозначить слева направо буквами A, B и C, то результат получится при выполнении перемножения $C \times (B \times A)$, или, поскольку операция умножения матриц ассоциативна, просто $C \times B \times A$). Для выполнения этой проверки вдумчивому читателю необходимо очень внимательно выписать матрицу B, поскольку первая мысль, которая может прийти в голову, скорее всего, неправильная.

Также читателю рекомендуется проверить, что данная схема делает с произвольной двухкубитовой системой, находящейся в некоторой суперпозиции четырёх указанных в первом столбце таблицы 5 базисных квантовых состояний [38].

А вот ещё один пример несложной квантовой схемы:

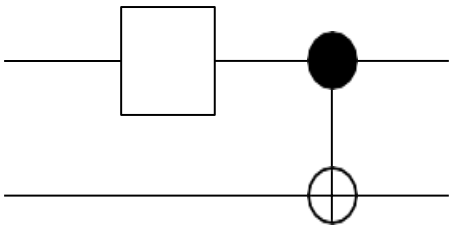


Рис. 2.11. Ещё один пример квантовой схемы

Эта квантовая схема получает на вход два кубита, которые запутывает между собой так, чтобы на выходе получилось одно из четырёх состояний Белла. Результат работы этой квантовой схемы для базисных состояний отражён в таблице 2.6:

Другими словами, квантовые схемы с точки зрения функционального программирования являются ничем иным, как *композицией* специального вида функций. Здесь название операции композиции выделено курсивом специально, поскольку это не совсем та композиция, которая известна в языке Haskell как оператор (.) — здесь имеет место последовательное выполнение унитарных преобразований (то есть в терминах функционального программирования — применение специального рода функций) над кубитами [39].

Таблица 2.6.

Результат работы квантовой схемы с рис. 2.11

Входные кубиты	Результат на выходе	Обозначение состояния Белла
$\|00\rangle$	$\frac{1}{\sqrt{2}}\|00\rangle + \frac{1}{\sqrt{2}}\|11\rangle$	$\|\Phi^+\rangle$
$\|01\rangle$	$\frac{1}{\sqrt{2}}\|01\rangle + \frac{1}{\sqrt{2}}\|10\rangle$	$\|\Psi^+\rangle$
$\|10\rangle$	$\frac{1}{\sqrt{2}}\|00\rangle - \frac{1}{\sqrt{2}}\|11\rangle$	$\|\Phi^-\rangle$
$\|11\rangle$	$\frac{1}{\sqrt{2}}\|01\rangle - \frac{1}{\sqrt{2}}\|10\rangle$	$\|\Psi^-\rangle$

Относительно квантовых схем осталось рассмотреть ещё один аспект. На двух предыдущих рисунках приведены квантовые схемы, в которых используются два кубита. Само собой разумеется, что в квантовых схемах может быть задействовано произвольное количество кубитов, и обычно весь набор кубитов называется *квантовым регистром*. Соответственно, некоторые унитарные операции действуют на отдельные кубиты в составе регистра, некоторые на часть, а некоторые гейты — на весь квантовый регистр в целом. В этом вопросе у разработчика алгоритма есть самая широкая свобода выбора.

Если на несколько кубитов в квантовом регистре одновременно действуют несколько же гейтов, то такие гейты можно объединять в один, действующий на все кубиты первоначальные сразу. Новый гейт получается просто как тензорное произведение всех гейтов, причём, поскольку тензорное произведение матриц некоммутативно, произведение начинается с самых «верхних» (на схеме) кубитов. Это положение следует из линейности модели квантовых вычислений.

Это положение можно пояснить при помощи следующей диаграммы:

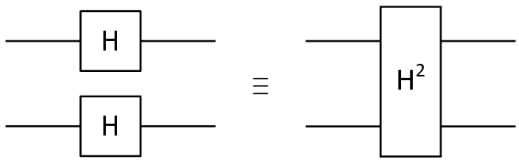


Рис. 2.12. Тождественность унитарных преобразований

На рис. 2.12 под меткой «H^2» имеется в виду тензорное произведение $H \otimes H$ (обычно записываемое как $H^{\otimes 2}$), которое, как и в случае векторов, вычисляется следующим образом. Пусть, как на вышеприведённом рисунке, есть два гейта Адамара, тогда тензорное произведение их матриц будет вычисляться как:

$$\begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix} \otimes \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix}$$

$$= \begin{pmatrix} \frac{1}{\sqrt{2}} \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix} & \frac{1}{\sqrt{2}} \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix} \\ \frac{1}{\sqrt{2}} \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix} & -\frac{1}{\sqrt{2}} \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix} \end{pmatrix}$$

$$= \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{pmatrix}$$

Данный гейт ($H^{\otimes 2}$) преобразовывает пару кубитов, подаваемых к нему на вход, из стандартного вычислительного базиса в базис $\{|+\rangle, |-\rangle\}$. Соответственно, гейт $H^{\otimes n}$ переводит в такой базис n кубитов в стандартном вычислительном базисе, представляя собой матрицу размеров $2^n \times 2^n$.

2.3. Квантовая схемотехника

Квантовая схемотехника — это, по сути, методология анализа и, главное, синтеза квантовых схем, реализующих те или иные алгоритмы (в общем понимании, не только квантовые). Обобщённо любой вычислительный процесс представляется в виде тройки (вход, процесс преобразования, выход). Принимая во внимание это соображение, задачами квантовой схемотехники можно назвать:

1. Прямой анализ. При наличии схемы входа и описания вычислительного процесса определить схему выхода.
2. Обратный анализ. При наличии описания вычислительного процесса и схемы выхода определить схему входа.
3. Синтез. При наличии схем входа и выхода построить описание вычислительного процесса.

К сожалению, в имеющейся литературе по квантовым вычислениям данные вопросы практически не находят своего отражения (от силы есть пара источников на русском языке, где кратко рассматриваются некоторые из них), а именно они являются краеугольным камнем прикладного программирования. Поэтому далее в этом разделе мы постараемся в полной мере раскрыть все три аспекта квантовой схемотехники.

Прямой анализ. С математической точки зрения эта задача решается достаточно просто. Входная схема — это описание множества возможных значений, которые могут быть поданы на вход квантовому вычислительному процессу. Поскольку квантовый регистр, то есть набор кубитов, может быть представлен в виде вектора, а каждый гейт в квантовой схеме представляет собой унитарную матрицу, то задача прямого анализа сводится к последовательному умножению матриц на вектор. А можно сначала перемножить все матрицы в правильном порядке, а затем итоговую матрицу умножить на входной вектор.

Проведя эту процедуру на всех возможных значениях входа (либо применив иные техники анализа), можно получить все выходные значения, после чего объединить их в схему. Данный вид анализа затрудняется тем, что при увеличении количества кубитов размерность векторов и матриц увеличивается экспоненциально. И если для квантовых регистров, состоящих из 1 — 3 кубитов такую процедуру ещё можно провести, то ручной анализ для четырёх кубитов уже затруднителен, а для десяти кубитов и

больше уже сложно проводить и перемножение матриц на компьютере.

Обратный анализ. В рамках модели квантовых вычислений задача обратного анализа тривиально сводится к задаче прямого анализа. Поскольку вычисления являются обратимыми, а матрицы всех гейтов — унитарными, то для обратного анализа заданной квантовой схемы достаточно обратить её, то есть перекоммутировать гейты в обратном порядке, сделав выход входом и наоборот, а сами гейты преобразовать в эрмитово-сопряжённые. После этого проводится описанная ранее процедура прямого анализа.

Само собой разумеется, что вышеприведённые рассуждения применимы только к квантовым схемам, в которых нет операции измерения, которая является необратимой. С другой стороны, измерение обычно применяется в самом конце квантовых вычислений, когда необходимо получить классический результат, поэтому обращение схемы можно осуществить, отбросив операции измерения. И есть совсем немного квантовых алгоритмов, в которых измерение производится в середине процесса вычислений (например, квантовая телепортация), и к таким алгоритмам и их квантовым схемам этот метод обратного анализа не подходит, и надо использовать иные (если они вообще существуют — в общем случае обратная задача неразрешима).

Синтез. Задача построения квантовой схемы по заданным входу и выходу усиливается, по сравнению с классическим случаем, необходимостью обеспечить обратимость вычислений. В общем случае произвольный вычислительный процесс может быть описан как двоичная функция, принимающая на вход n битов и возвращающая m битов. Такая функция в классическом варианте может быть построена при помощи базисного (универсального) набора логических элементов.

Наличие классической схемы из универсального набора строительных блоков для заданной функции переводит задачу синтеза в задачу построения квантового оракула. Как уже было указано выше, квантовый оракул строится из классического выражения функции следующим образом:

1. Квантовый оракул будет иметь по $(n + m)$ входов и выходов. Первые n входов принимают входные параметры функции, а следующие m входов инициализируются в $|0\rangle$. Соответственно, первые n выходов возвращают входные данные, а следующие m

выходов получают сумму по модулю 2 (операция Исключающее ИЛИ) значения функции на заданном входе с m входными данными.

2. Все элементы классической схемы преобразуются в соответствующие квантовые гейты (например, можно воспользоваться универсальным набором из гейтов *H* и *CNOT*, либо квантовым аналогом элемента Тоффоли). Это может привести к появлению огромного количества так называемых мусорных кубитов.

3. Построенная схема из универсальных квантовых элементов подвергается процессу оптимизации. Это довольно нетривиальный процесс, и в каждом случае применяются свои методы и техники. Цель этого процесса — свести количество мусорных кубитов к минимуму, желательно вообще их исключить.

Если от мусорных кубитов избавиться не удалось (пусть их осталось l), то все они добавляются к входным и выходным (стало быть, входов и выходов становится $(n + m + l)$). Но эти кубиты должны быть вначале инициализированы в $|0\rangle$, а по окончании вычислений они также должны быть установлены в $|0\rangle$. Это следствие законов квантовой механики.

Таким образом, будет построена квантовая схема классической функции. Однако, это не единственный способ. Другим вариантом является построение одной унитарной матрицы для представления полной классической функции. Эта задача может быть успешно разрешена при помощи решения системы уравнений, получаемой из произведения матрицы на вектор. Проблема лишь в том, что количество уравнений и неизвестных растёт экспоненциально от количества кубитов. И если количество входов и выходов равно $(n + m)$, то количество уравнений и неизвестных в них будет равно, как полагается, $2^{2(n + m)}$. Решать такую систему просто для небольших чисел, а вот для больших уже проблематично.

В качестве примера можно рассмотреть построение квантового аналога классической логической операции И. Функция, реализующая эту операцию, имеет 2 входа и 1 выход, следовательно у квантового оракула будет по 3 входа и выхода. Следующая таблица показывает 8 возможных вариантов поведения квантового оракула.

Таблица 2.7.

Входы и выходы квантового оракула
для операции И

X_1	X_2	Y	$X_1 \& X_2$	$(X_1 \& X_2) \oplus Y$
0	0	0	0	0
0	0	1	0	1
0	1	0	0	0
0	1	1	0	1
1	0	0	0	0
1	0	1	0	1
1	1	0	1	1
1	1	1	1	0

Следовательно, система уравнений, которую надо решить, выглядит следующим образом:

$$I \times \begin{pmatrix} \alpha_{11} & \alpha_{12} & \alpha_{13} & \alpha_{14} & \alpha_{15} & \alpha_{16} & \alpha_{17} & \alpha_{18} \\ \alpha_{21} & \alpha_{22} & \alpha_{23} & \alpha_{24} & \alpha_{25} & \alpha_{26} & \alpha_{27} & \alpha_{28} \\ \alpha_{31} & \alpha_{32} & \alpha_{33} & \alpha_{34} & \alpha_{35} & \alpha_{36} & \alpha_{37} & \alpha_{38} \\ \alpha_{41} & \alpha_{42} & \alpha_{43} & \alpha_{44} & \alpha_{45} & \alpha_{46} & \alpha_{47} & \alpha_{48} \\ \alpha_{51} & \alpha_{52} & \alpha_{53} & \alpha_{54} & \alpha_{55} & \alpha_{56} & \alpha_{57} & \alpha_{58} \\ \alpha_{61} & \alpha_{62} & \alpha_{63} & \alpha_{64} & \alpha_{65} & \alpha_{66} & \alpha_{67} & \alpha_{68} \\ \alpha_{71} & \alpha_{72} & \alpha_{73} & \alpha_{74} & \alpha_{75} & \alpha_{76} & \alpha_{77} & \alpha_{78} \\ \alpha_{81} & \alpha_{82} & \alpha_{83} & \alpha_{84} & \alpha_{85} & \alpha_{86} & \alpha_{87} & \alpha_{88} \end{pmatrix}$$

$$= \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

Её решение тривиально, и результатом будет следующая матрица:

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

Эта матрица и есть квантовый оракул в матричном представлении. Теперь необходимо убедиться, что она унитарна. Для этого надо умножить её на эрмитово-сопряжённую. Поскольку эта матрица является самосопряжённой, то её надо просто возвести в квадрат. Несложные умозаключения дают понять, что результатом будет единичная матрица, то есть найденная матрица является унитарной.

Краткие пояснения. Единичная матрица в исходном уравнении получается из восьми строк таблицы 8, в которых выделены столбцы 1, 2 и 3. Каждая из этих восьми строк преобразуется в вектор-столбец (здесь ради экономии показаны векторыстроки, но надо помнить, что это векторы-столбцы). Строка «0 0 0» представляет собой вектор (1 0 0 0 0 0 0 0), строка «0 0 1» представляет вектор (0 1 0 0 0 0 0 0) и т. д. до строки «1 1 1», которая есть вектор (0 0 0 0 0 0 0 1). Результирующая матрица строится по тому же принципу, однако теперь из таблицы 8 берётся строки со столбцами 1, 2 и 5. Как видно, столбец 5 отличается от столбца 3 только в 7-ой и 8-ой строке, отсюда и перемена 7-го и 8-го столбцов в результирующей матрице.

Другими словами, для получения результирующей матрицы квантового оракула достаточно собрать векторы-столбцы по тому же принципу, как это описано в предыдущем абзаце. Для n входных кубитов берётся 2^n строк, в каждой из которых $(n + 1)$ столбец. Первые n столбцов принимают все возможные значения входных кубитов, а последний столбец — значение $(y \oplus f(\bar{x}))$. Для каждой строки ставится в соответствие векторстолбец в вычислительном базисе, а потом все столбцы собираются в матрицу.

А можно воспользоваться и более простым методом. Для этого каждой строке матрицы сопоставляется входной кубит (сверху вниз), а каждому столбцу матрицы сопоставляется выходной кубит (слева направо). В самой матрице значение 1 ставится на тех местах,

которые обозначают преобразование входного кубита в выходной, а на остальных позициях ставится значение 0. Для быстрого построения оракулов этот метод проще всего. Ну и можно при желании доказать его тождественность предыдущему методу.

Задача решена. Хорошим упражнением для заинтересованного читателя будет построение таких же матриц для других пятнадцати двоичных функций от двух переменных.

При всём описанном необходимо понимать, что синтез квантовой схемы — это не построение квантового алгоритма. Показанная здесь техника позволяет решать на квантовом компьютере произвольную вычислительную задачу. А разработка квантового алгоритма — это дело настолько необычное и нетривиальное, что чаще помогает озарение, нежели скрупулёзное построение шаг за шагом. На текущий день все разработанные квантовые алгоритмы базируются на нескольких базовых, которые были найдены именно при помощи озарения.

2.4.Принципы квантовых вычислений

Теперь осталось перейти к перечислению основополагающих принципов, на которых зиждется модель квантовых вычислений. Знание и отчётливое понимание сути этих принципов позволит в дальнейшем легко и бегло знакомиться с более глубокой литературой по квантовым вычислениям.

Обратимость вычислений — главный принцип, из которого, в общем-то, следуют все остальные. Квантовые вычисления обратимы во времени, а это значит, что по результату любого вычисления можно восстановить исходные данные. Более того, для любой квантовой схемы можно построить дуальную схему, которая будет получать на вход результат работы первоначальной схемы и возвращать исходные данные для неё. Для этого достаточно всего лишь перевернуть схему справа налево, а все унитарные преобразования заменить на эрмитово-сопряжённые.

Поскольку все унитарные преобразования являются обратимыми, все они имеют ровно столько же выходов, сколько и входов — не меньше, но и не больше (если бы было больше, то эрмитово-сопряжённое преобразование имело бы меньше выходов, а, значит, теряло бы информацию и, стало быть, не было бы обратимым).

Из этих рассуждений следует, что квантовые вычисления страшно *избыточны*. Ведь, например, самые простые логически

обратимые элементы Тоффоли и Фредкина имеют по три входа и три выхода. Если эти элементы используются для получения, скажем, элемента НЕ, то по два входа и выхода будут незадействованными. А для элементов И, ИЛИ и других неиспользуемыми будет один вход и два выхода. Для элемента FANOUT наоборот два входа использоваться не будут, равно как и один выход.

Это можно проиллюстрировать следующей диаграммой. На рис. 2.13 показано выражение элемента Тоффоли через элемент Фредкина. Из всего огромного множества входных и выходных битов используются только по три с каждой стороны.

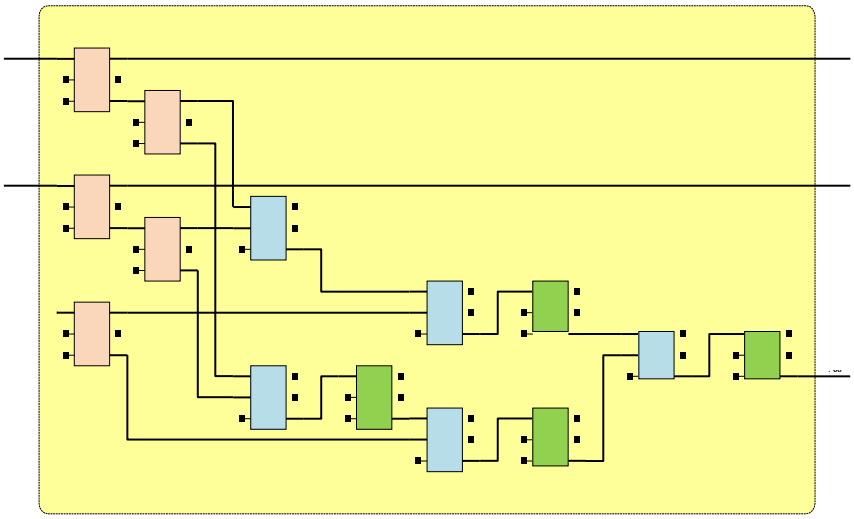


Рис. 2.13. Диаграмма, показывающая один из возможных вариантов выражения элемента Тоффоли через элемент Фредкина (красные элементы — FANOUT, синие — И, зелёные — НЕ)

Данный принцип влечёт, что в процессе работы квантовых алгоритмов будет накапливаться огромное количество «мусорных» битов. Если их начать уничтожать (например, при помощи своеобразной «сборки мусора»), то начнёт выделяться огромное количество теплоты, а вычисления станут необратимыми.

Собственно, на рис. 14 видно, что для выражения одного элемента с тремя входами и тремя выходами по обратимой схеме пришлось задействовать 14 базовых элементов, и общее число входов и выходов на схеме стало равным по 26, из которых

только по 3 входа и выхода используются, а остальные являются вспомогательными и, по сути, «мусорными».

Есть мнение, что за экспоненциальное ускорение решения некоторых задач, которое даёт модель квантовых вычислений, придётся заплатить экспоненциальным увеличением размера задействованной памяти, и ситуация с выражением элемента Тоффоли через элемент Фредкина это отчасти иллюстрирует. Впрочем, для решения этой проблемы был разработан метод, который позволяет уничтожать промежуточные «мусорные» биты без потери обратимости вычислений. Метод этот математически достаточно прост, и заинтересованный читатель сможет найти его описание в специализированной литературе.

Однако, конечно же, при более вдумчивом подходе, когда инженер или программист включает своё воображение и техническую смётку, можно получить более экономные способы выражения одних элементов через другие. В частности, рассмотренный пример можно преобразовать таким образом, что для выражения элемента Тоффоли через элементы Фредкина потребуется всего 4 последних. Из этих четырёх элементов формируется схема, в которой есть 6 выходов, однако все они могут быть задействованы. Три выхода являются непосредственным решением задачи, а три других могут использоваться в дальнейшем, поскольку тем или иным способом преобразуют входные биты. Вдумчивому читателю предлагается самостоятельно найти такую схему выражения элемента Тоффоли (это просто).

Всё это обозначает, что в квантовых схемах *нет и не может быть циклов и возвратов назад*. Кубиты как бы двигаются по «проводам», проходя через гейты и преобразовываясь в соответствии с предназначением гейта. Поток управления квантовой программой движется от начала алгоритма только вперёд. Унитарное преобразование и контролируемое унитарное преобразование — вот единственные способы выполнения вычислений. Измерение — единственный способ получения результата, который, к тому же, «схлопывает» суперпозицию, в которой находятся кубиты.

По поводу обратимых вычислений остаётся отметить, что одним из немаловажных аспектов является обработка ненужных выходов. Если просто уничтожать результаты их вычислений, то такое обращение ничем не будет отличаться от обычной классической

вычислительной модели, поскольку при уничтожении информации выделяется тепло (энергия, задействованная в процессе хранения и передачи информации рассеивается). Выделение тепла делает обратимые вычисления необратимыми, а саму модель квантовых вычислений бесполезной. Поэтому при разработке квантовых схем используются специальные методы, которые собирают все неиспользованные выходы и преобразуют их специальным образом так, чтобы они использовались (например, для вычисления обратной функции), а весь процесс вычислений и его квантовая схема были полностью обратимыми.

Следующим принципом является *квантовый параллелизм*. Система кубитов в процессе прохождения через гейты унитарных преобразований параллельно решает одну и ту же задачу для огромного, экспоненциально большого количества данных из-за того, что кубиты находятся в суперпозиции базисных состояний. Поскольку с ростом числа кубитов размерность базиса растёт в степенной зависимости, квантовый параллелизм обладает огромной вычислительной мощью. Главное — уметь правильно пользоваться этим принципом.

С параллелизмом тесно связана *интерференция*. Некоторые алгоритмы используют интерференцию квантовых состояний для того, чтобы взаимно усилить требуемый результат и наоборот сгладить результат нежелательный. Повторяя несколько раз последовательность параллельной обработки кубитов с интерференцией их состояний, разработчик алгоритма может так усилить амплитуду искомого состояния, что дальнейшее измерение даст требуемый результат с высокой вероятностью (варьируя количество повторений шага с интерференцией, можно управлять уровнем вероятности, доводя его до любого заданного значения).

Квантовая запутанность — ещё один принцип, причём наименее изученный, равно как и наименее поддающийся рациональному осмыслению. Учёные уже больше века ломают головы и копья на тему запутанности квантовых систем (взять, хотя бы, спор Н. Бора с А. Эйнштейном на эту тему). Но именно квантовая запутанность позволяет решить многие хитрые задачи, являясь ключевым фактором многих квантовых алгоритмов [34-37].

Общая архитектура квантового компьютера

Было бы странным полагать, что квантовые компьютеры, когда они будут реализованы в «железе», будут представлять собой некие устройства, похожие на компьютеры современные. Уже тот принцип, который говорит о недопустимости циклов и возвратов потока управления программой, намекает на то, что квантовый компьютер будет каким-то иным устройством. А есть ещё взаимодействие с пользователем, работа с периферийными устройствами, работа по сети с другими компьютерами — всё это вряд ли возможно будет реализовать в рамках модели квантовых вычислений, да и не будет такой необходимости, поскольку классические компьютеры очень неплохо справляются с этими задачами.

Скорее всего, все разработанные и реализованные к моменту создания квантового компьютера алгоритмы будут упакованы в некий «Фонд алгоритмов и программ», библиотеку квантовых алгоритмов, которая будет связана с устройством, выполняющим квантовые вычисления. Вполне вероятно, что это будет некий «квантовый сопроцессор», которым управляет обычный процессор при помощи каких-либо хитроумных устройств, которые позволяют выполнять две основные задачи: подготовка необходимых унитарных преобразований для выбранного квантового алгоритма и инициализация входных кубитов. На выходе этого «квантового сопроцессора» будут производиться измерения, а их результат сообщаться основному процессору, производящему вычисления.

Так что от взаимодействия с внешним миром, циклов и возвратов назад по некоторым условиям никуда не деться — всё это очень мощные идиомы программирования, чтобы от них отказываться. Но выполнять эти операции будет классический компьютер, а квантовому сопроцессору останется только по управляющему вызову из основного вычислительного устройства запускать определённый алгоритм из библиотеки и возвращать результат его работы.

Всё это позволяет нарисовать примерно следующую общую архитектуру, в рамках которой будут производиться квантовые вычисления [39]:

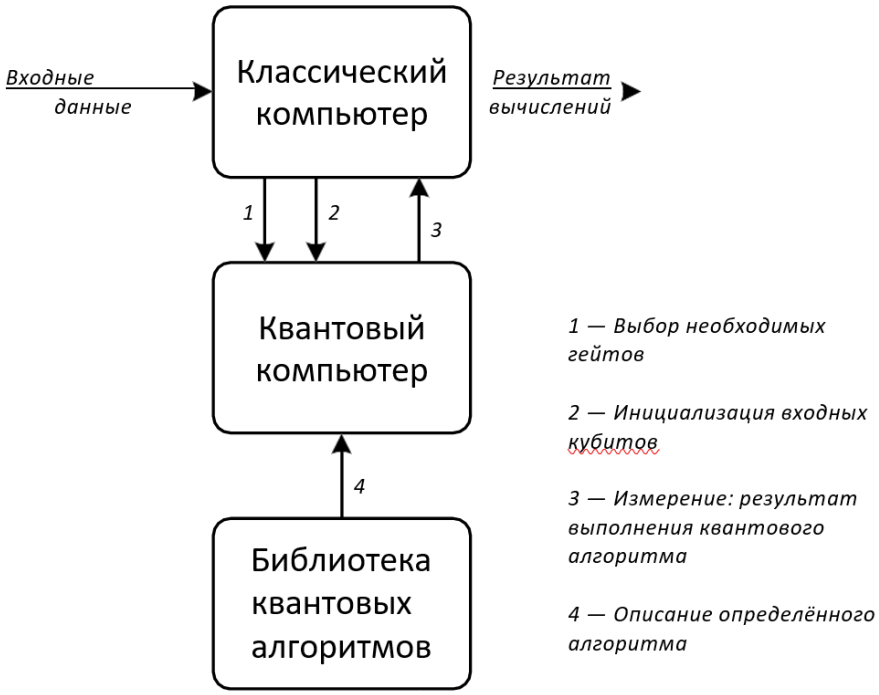


Рис. 2.14. Общая архитектура квантового компьютера

Исходя из этой диаграммы, можно дальше попытаться пофантазировать на тему того, как будет развиваться отрасль квантового программирования. Оставив в стороне «физиков», которые занимаются сейчас и будут заниматься в будущем поиском методов реализации квантового вычислительного устройства в «железе», можно разделить программистов на два больших класса:

- *Системные квантовые программисты* будут создавать новые изощрённые квантовые алгоритмы для размещения их в библиотеке квантовых алгоритмов и последующего использования в работе всей системы.
- *Прикладные квантовые программисты* будут использовать квантовые алгоритмы на практике, решая при помощи них прикладные задачи, но при этом они должны будут знать как номенклатуру алгоритмов в библиотеке, так и понимать смысл их работы, чтобы оптимальным образом производить их вызов и использование их результатов в прикладных программах для обычного компьютера.

Таким образом, программировать само квантовое вычислительное устройство никто не будет; его будет «программировать» классический компьютер — настраивать гейты, связывать их друг с другом, инициализировать кубиты и запускать

квантовые вычисления. Как именно это будет реализовано, сейчас пока ещё никто не знает.

Модель квантовых вычислений — это некоторая математическая система, которая позволяет производить произвольные вычисления, причём уже показано, что для некоторых задач эта модель позволяет реализовать намного более эффективные алгоритмы, чем традиционная вычислительная модель, основанная на машине Тьюринга или лямбда-исчислении. Реализация этой новой вычислительной модели в «железе» не имеет фундаментальных преград, так что дело только за технологиями.

Сама модель сводится к выполнению ряда унитарных преобразований векторов в 2^n-мерном гильбертовом пространстве, где n — количество кубитов, а каждое унитарное преобразование представляет собой матрицу специального вида. Таким образом, очень упрощённо, можно сказать, что по своей сути квантовые вычисления заключаются в перемножении матрицы размера $2^n \times 2^n$ (которая сама ещё должна быть получена как произведение целого набора матриц такого же или меньших размеров) на вектор размера 2^n. Эти вычисления, конечно же, можно выполнить и на обычных компьютерах, но сложность такого умножения будет расти экспоненциально в зависимости от количества моделируемых кубитов.

Квантовые алгоритмы

В квантовых вычислениях квантовый алгоритм – это алгоритм, который работает на реалистичной модели квантовых вычислений. Представляет собой последовательность унитарных операций (гейтов) с указанием, над какими именно кубитами их надо совершать.

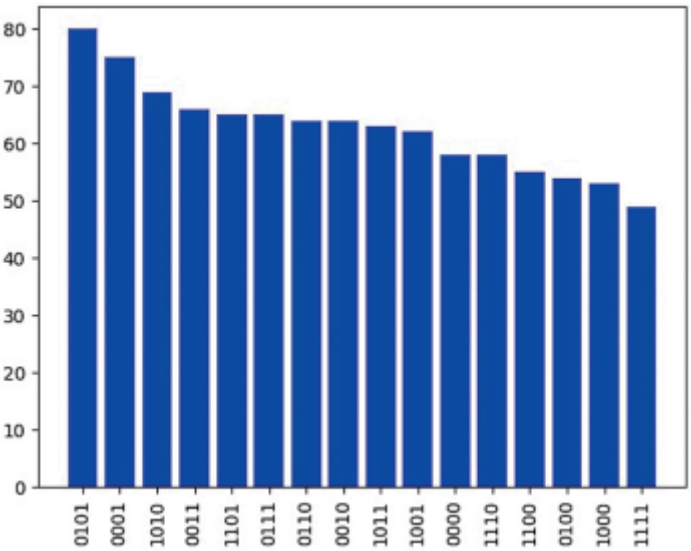


Рис. 2.15. Результат работы программной модели квантовых вычислений

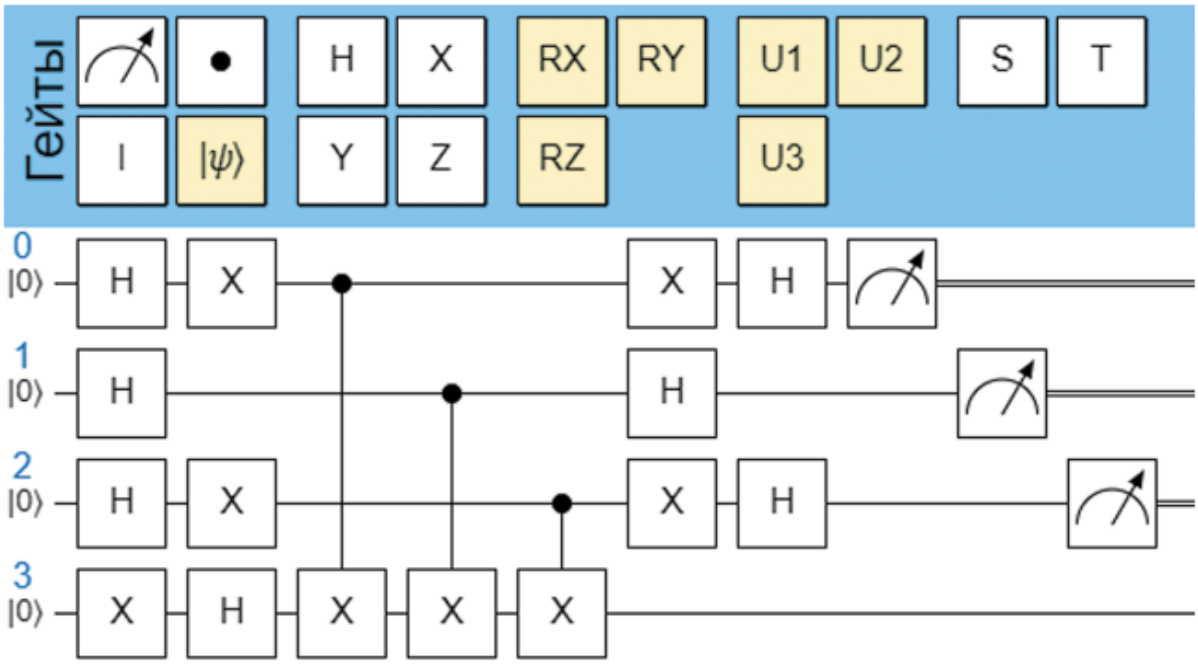


Рис. 2.16. Квантовая схема алгоритма Дойча – Йожи

В рамках данной работы была смоделирована работа набора квантовых алгоритмов, описание некоторых из них приведено ниже.

Алгоритм Дойча – Йожи

Известно, что скрытая булева функция, которая принимает в качестве входных данных строку битов и возвращает либо 0, либо 1, является постоянной или сбалансированной. Необходимо определить

тип функции, обратившись к вычисляющему функцию оракулу наименьшее возможное число раз. Квантовая схема алгоритма представлена на рисунке 2.16.

Алгоритм представляет собой применение к нулевому вектору $|0\rangle^{\otimes n}$ оператора

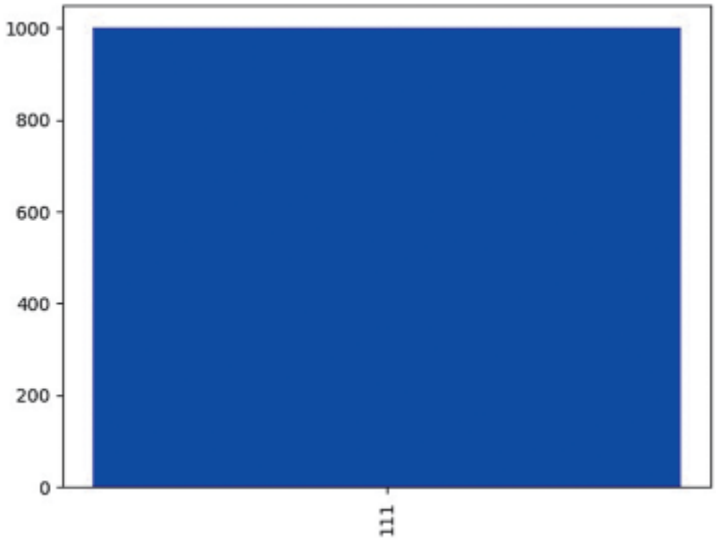


Рис. 2.17. Результат работы алгоритма Дойча – Йожи

$H^{\otimes n}U_f H^{\otimes n}$, где H – это гейт Адамара, U_f – квантовый оракул, а $\otimes n$ – возведение в степень n через тензорное произведение. Если после измерения регистра все биты равны 0, значит значение функции $f(x)$ не зависит от x, в противном случае функция является сбалансированной. Для данного алгоритма можно создать квантовый оракул путем применения гейтов CNOT с каждым входным кубитом в качестве элемента управления и выходным битом в качестве цели. Изначальные состояния, которые дают 0 или 1, можно варьировать путем применения гейта X к некоторым элементам управления. Затем необходимо перевести входные кубиты в состояние $|+\rangle$, а выходной кубит в состояние $|-\rangle$. В симуляторе это можно сделать при помощи гейтов (гейтов Адамара и гейта NOT), а можно при помощи изменения базисного состояния кубита. После применения оракула необходимо использовать гейты Адамара на входных кубитах, а

затем измерить их. Результат работы алгоритма приведен на рисунке 2.17.

Данный результат говорит о том, что функция сбалансирована.

Алгоритм Бернштейна – Вазирани

Алгоритм Бернштейна – Вазирани является расширенной версией алгоритма Дойча – Йожи. Известно, что существует скрытая булева функция, которая принимает в качестве входных данных строку битов и возвращает либо 0, либо 1, причем $f(x)=a\cdot x$, где $\cdot$ – скалярное произведение.

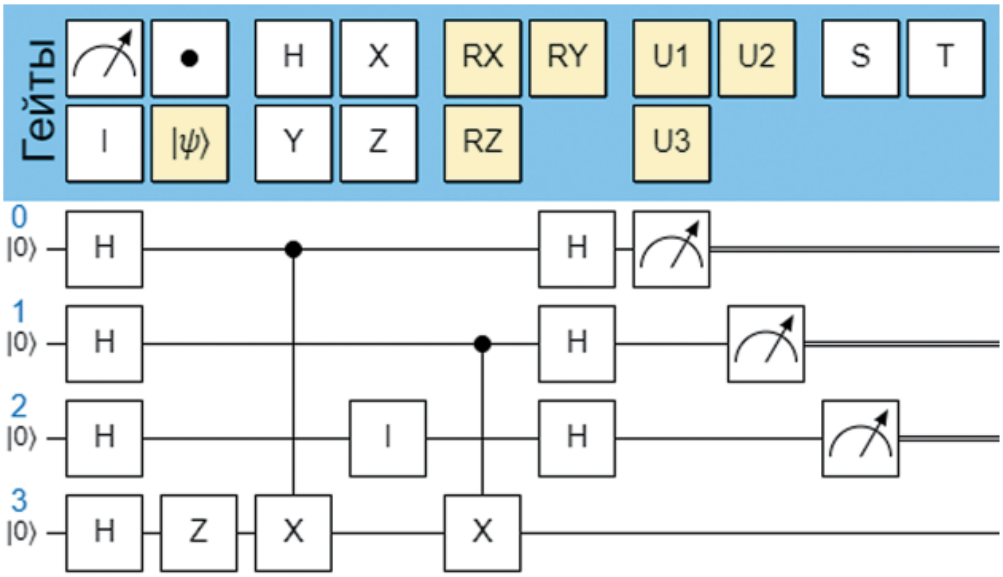


Рис. 2.18. Квантовая схема алгоритма Бернштейна – Вазирани

Требуется найти a. Квантовая схема алгоритма с $a=011$ представлена на рисунке 2.18. Алгоритм Бернштейна – Вазирани демонстрирует преимущество квантовых алгоритмов по наименьшему требуемому количеству запросов к оракулу. Классическое решение задачи в лучшем случае потребует $O(n)$ обращений к оракулу, в то время как квантовый алгоритм решает задачу за $O(1)$ [8].

Основные кубиты инициализируются состоянием $|0\rangle^{\otimes n}$, а вспомогательный состоянием $|1\rangle$. Затем ко всем кубитам применяется гейт Адамара. К результатам предыдущего шага применяется оракул

U_f, после чего к основным кубитам снова применяется гейт Адамара. В результате измерение входного регистра даст искомый «скрытый» вектор. Результат работы алгоритма приведен на рисунке 2.19. Результатом измерения является искомый «скрытый» вектор – 011.

Квантовое преобразование Фурье

В классических вычислениях преобразование Фурье – это математическая операция, которая является фундаментальной для таких областей, как обработка сигналов. Квантовое преобразование Фурье (QFT) выполняет преобразование между двумя базисами, вычислительным *(Z)* базисом и базисом Фурье, и является частью некоторых важных алгоритмов, таких как алгоритм Шора [9]. Гейт Адамара – это однокубитное QFT, так как он выполняет преобразование из базиса $Z|0\rangle$ и $|1\rangle$ в базис $X|+\rangle$ и $|-\rangle$. Точно так же все многокубитные состояния в вычислительном базисе имеют соответствующие состояния в базисе Фурье, то есть QFT – это просто функция, которая переводит состояние из одного базиса в другой.

Схема квантового преобразования Фурье для двух кубитов приведена на рисунке 2.19.

В многокубитных (2 и больше) квантовых схемах, реализующих QFT, используются два гейта: гейт Адамара и контролируемый фазовый поворот.

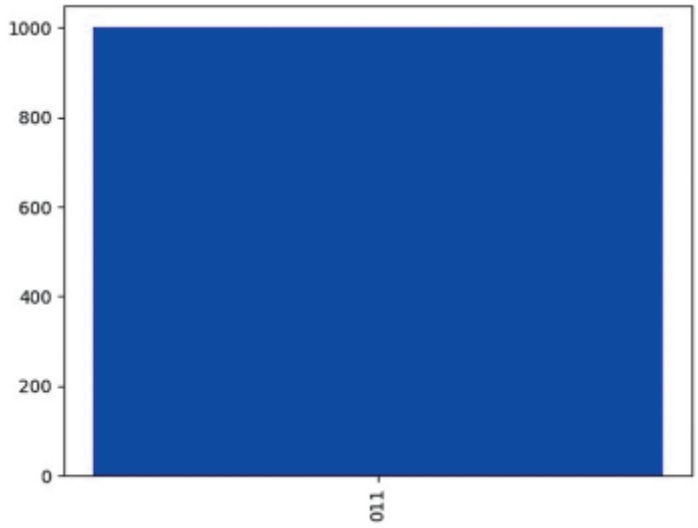


Рис. 2.19. Результат работы алгоритма Бернштейна – Вазирани

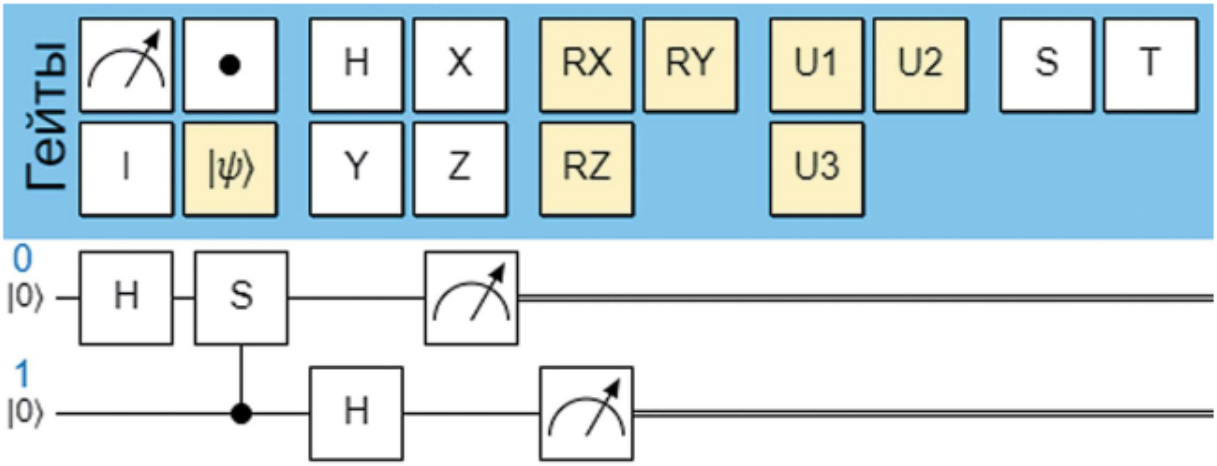


Рис. 2.20. Квантовая схема QFT для двух кубитов

Для 2-кубитного QFT к первому кубиту сначала применяется гейт Адамара,

который переводит кубит в суперпозицию.

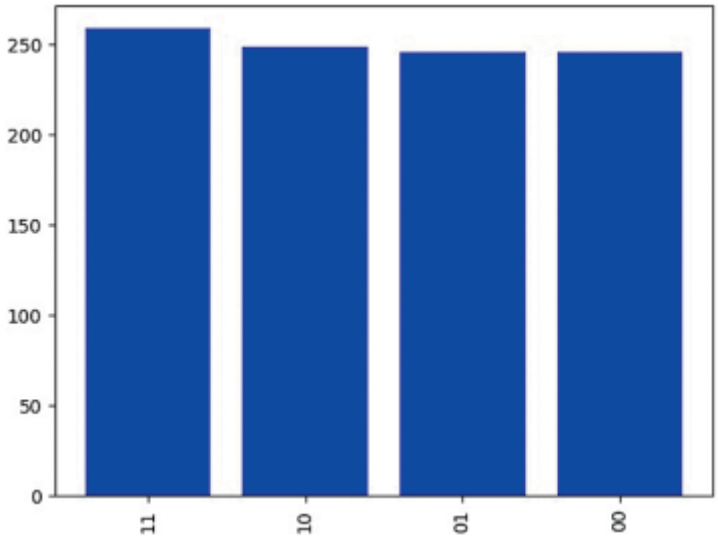


Рис. 2.21. Результат работы схемы квантового преобразования Фурье для 2 кубитов

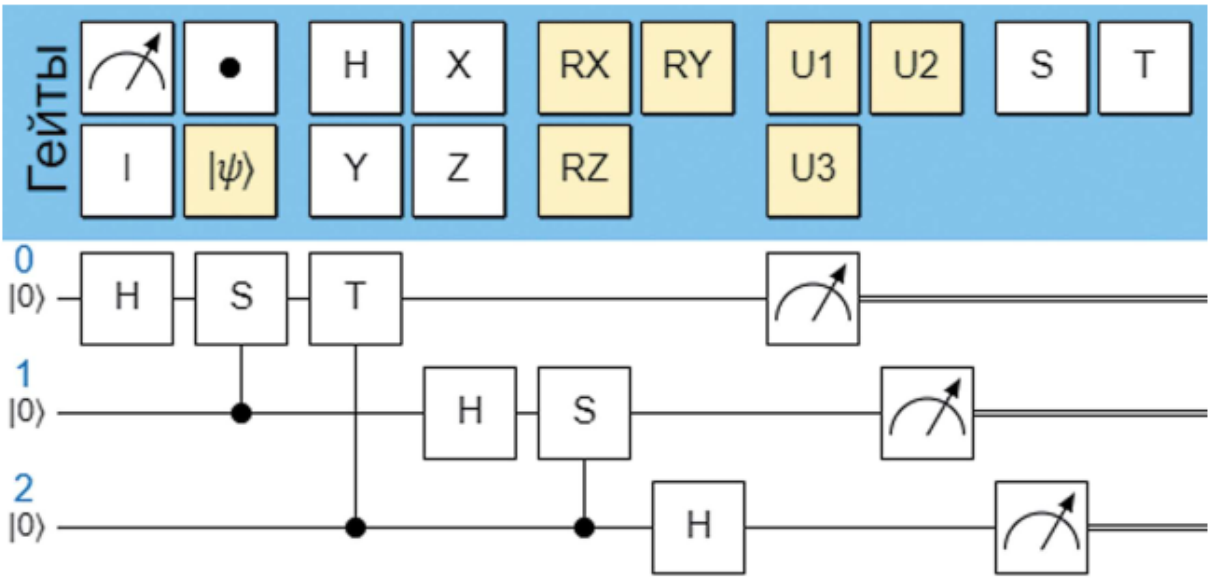


Рис. 2.22. Квантовая схема QFT для трех кубитов

Затем к кубиту применяется контролируемый гейт S (являющийся частным случаем гейта $U1$ с $\lambda = \pi/2$). После этого ко второму кубиту применяется гейт Адамара. Результат работы квантовой схемы приведен на рисунке 2.21.

Схема квантового преобразования Фурье для трех кубитов приведена на рисунке 2.22.

С увеличением количества кубитов алгоритм усложняется. QFT для 3 кубитов требует выполнения следующих шагов:

1) Применить гейт Адамара к первому кубиту $|x_1\rangle$;

2) К первому кубиту применить гейт S, контролируемый вторым кубитом;

3) Применить к первому кубиту гейт T (являющийся частным случаем гейта $U1$ с $\lambda = \pi/4$), контролируемый вторым кубитом;

4) Применить гейт Адамара ко второму кубиту;

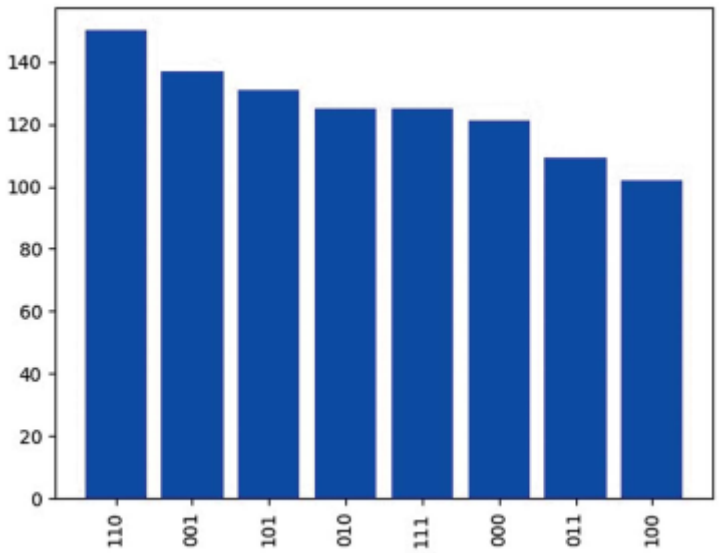


Рис. 2.23. Результат работы схемы квантового преобразования Фурье для 3 кубитов

5) Применить гейт S, контролируемый третьим кубитом, ко второму кубиту;

6) Применить гейт Адамара к третьему кубиту.

Результат работы квантовой схемы приведен на рисунке 2.23.

Схема квантового преобразования Фурье для четырех кубитов приведена на рисунке 2.14.

4-кубитная схема, реализующая QFT, построена по тому же принципу, что и схема для 3 кубитов, но в ней используется гейт поворота $U1$ с $\lambda = \pi/8$. Соответственно в 5-кубитной схеме используется $U1$ с $\lambda = \pi/16$, в 6-кубитной $U1$ с $\lambda = \pi/32$ и т.д. Результат работы квантовой схемы приведен на рисунке 2.15.

Реализация квантового компьютера позволит решить проблему экспоненциального роста сложности, при этом такой компьютер будет гибридным — скорее всего, аналоговое квантовое вычислительное устройство будет управляться обычным цифровым компьютером.

Глава 3. ПОДХОДЫ К РЕШЕНИЮ ЗАДАЧ ОПТИМИЗАЦИИ

3.1. Анализ решения задач оптимизации классическими методами

При решении конкретной задачи оптимизации исследователь прежде всего должен выбрать математический метод, который приводил бы к конечным результатам с наименьшими затратами на вычисления или же давал возможность получить наибольший объем информации об искомом решении. Выбор того или иного метода в значительной степени определяется постановкой оптимальной задачи, а также используемой математической моделью объекта оптимизации.

В настоящее время для решения оптимальных задач применяют в основном следующие методы [1,26,28]:

1) методы исследования функций классического анализа;
2) методы, основанные на использовании неопределенных множителей Лагранжа;
3) вариационное исчисление;
4) динамическое программирование;
5) принцип максимума;
6) линейное программирование;
7) нелинейное программирование.

Наилучшим путем при выборе метода оптимизации, наиболее пригодного для решения соответствующей задачи, следует признать исследование возможностей и опыта применения различных методов оптимизации. Отметим также, что некоторые методы специально разработаны или наилучшим образом подходят для решения оптимальных задач с математическими моделями определенного вида. Так, математический аппарат линейного программирования, специально создан для решения задач с линейными критериями оптимальности и линейными ограничениями на переменные и позволяет решать большинство задач, сформулированных в такой постановке. Так же и геометрическое программирование предназначено для решения оптимальных задач, в которых критерий оптимальности и ограничения представляются специального вида функциями полиномами.

Динамическое программирование хорошо приспособлено для решения задач оптимизации многостадийных процессов, особенно

тех, в которых состояние каждой стадии характеризуется относительно небольшим числом переменных состояния. Однако при наличии значительного числа этих переменных, т. е. при высокой размерности каждой стадии, применение метода динамического программирования затруднительно вследствие ограниченных быстродействия и объема памяти вычислительных машин [28,40].

Ниже приводится краткий обзор математических методов решения оптимальных задач и примеры их использования. Здесь же дана лишь краткая характеристика указанных методов и областей их применения, что до некоторой степени может облегчить выбор того или иного метода для решения конкретной оптимальной задачи.

Методы исследования функций классического анализа представляют собой наиболее известные методы решения несложных оптимальных задач, с которыми известны из курса математического анализа. Обычной областью использования данных методов являются задачи с известным аналитическим выражением критерия оптимальности, что позволяет найти не очень сложное, также аналитическое выражение для производных. Полученные приравниванием нулю производных уравнения, определяющие экстремальные решения оптимальной задачи, крайне редко удается решить аналитическим путем, поэтому, как, правило, применяют вычислительные машины. При этом надо решить систему конечных уравнений, чаще всего нелинейных, для чего приходится использовать численные методы, аналогичные методам нелинейного программирования [26,28,94].

Дополнительные трудности при решении оптимальной задачи методами исследования функций классического анализа возникают вследствие того, что система уравнений, получаемая в результате их применения, обеспечивает лишь необходимые условия оптимальности. Поэтому все решения данной системы (а их может быть и несколько) должны быть проверены на достаточность. В результате такой проверки сначала отбрасывают решения, которые не определяют экстремальные значения критерия оптимальности, а затем среди остающихся экстремальных решений выбирают решение, удовлетворяющее условиям оптимальной задачи, т. е. наибольшему или наименьшему значению критерия оптимальности в зависимости от постановки задачи.

Методы исследования при наличии ограничений на область изменения независимых переменных можно использовать только для

отыскания экстремальных значений внутри указанной области. В особенности это относится к задачам с большим числом независимых переменных (практически больше двух), в которых анализ значений критерия оптимальности на границе допустимой области изменения переменных становится весьма сложным [26,28,121].

Метод множителей Лагранжа применяют для решения задач такого же класса сложности, как и при использовании обычных методов исследования функций, но при наличии ограничений типа равенств на независимые переменные. К требованию возможности получения аналитических выражений для производных от критерия оптимальности при этом добавляется аналогичное требование относительно аналитического вида уравнений ограничений.

В основном при использовании метода множителей Лагранжа приходится решать те же задачи, что и без ограничений. Некоторое усложнение в данном случае возникает лишь от введения дополнительных неопределенных множителей, вследствие чего порядок системы уравнений, решаемой для нахождения экстремумов критерия оптимальности, соответственно повышается на число ограничений. В остальном, процедура поиска решений и проверки их на оптимальность отвечает процедуре решения задач без ограничений [19].

Множители Лагранжа можно применять для решения задач оптимизации объектов на основе уравнений с частными производными и задач динамической оптимизации. При этом вместо решения системы конечных уравнений для отыскания оптимума необходимо интегрировать систему дифференциальных уравнений.

Следует отметить, что множители Лагранжа используют также в качестве вспомогательного средства и при решении специальными методами задач других классов с ограничениями типа равенств, например, в вариационном исчислении и динамическом программировании. Особенно эффективно применение множителей Лагранжа в методе динамического программирования, где с их помощью иногда удается снизить размерность решаемой задачи.

Методы вариационного исчисления обычно используют для решения задач, в которых критерии оптимальности представляются в виде функционалов и решениями которых служат неизвестные функции. Такие задачи возникают обычно при статической оптимизации процессов с распределенными параметрами или в задачах динамической оптимизации.

Вариационные методы позволяют в этом случае свести решение оптимальной задачи к интегрированию системы дифференциальных уравнений Эйлера, каждое из которых является нелинейным дифференциальным уравнением второго порядка с граничными условиями, заданными на обоих концах интервала интегрирования. Число уравнений указанной системы при этом равно числу неизвестных функций, определяемых при решении оптимальной задачи. Каждую функцию находят в результате интегрирования получаемой системы.

Уравнения Эйлера выводятся как необходимые условия экстремума функционала. Поэтому полученные интегрированием системы дифференциальных уравнений функции должны быть проверены на экстремум функционала.

При наличии ограничений типа равенств, имеющих вид функционалов, применяют множители Лагранжа, что дает возможность перейти от условной задачи к безусловной. Наиболее значительные трудности при использовании вариационных методов возникают в случае решения задач с ограничениями типа неравенств.

Заслуживают внимания прямые методы решения задач оптимизации функционалов, обычно позволяющие свести исходную вариационную задачу к задаче нелинейного программирования, решить которую иногда проще, чем краевую задачу для уравнений Эйлера.

3.1.1.Классификация математической модели оптимизации

Математическое программирование – это математическая дисциплина, в которой разрабатываются методы отыскания экстремальных значений целевой функции среди множества ее возможных значений, определяемых ограничениями.

Наличие ограничений делает задачи математического программирования принципиально отличными от классических задач математического анализа по отысканию экстремальных значений функции. Методы математического анализа для поиска экстремума функции в задачах математического программирования оказываются непригодными.

Для решения задач математического программирования разработаны и разрабатываются специальные методы и теории. Так как при решении этих задач приходится выполнять значительный

объем вычислений, то при сравнительной оценке методов большое значение придается эффективности и удобству их реализации на ЭВМ.

Математическое программирование можно рассматривать как совокупность самостоятельных разделов, занимающихся изучением и разработкой методов решения определенных классов задач.

В зависимости от свойств целевой функции и функции ограничений все задачи математического программирования делятся на два основных класса:

- задачи линейного программирования,
- задачи нелинейного программирования.

Если целевая функция и функции ограничений – линейные функции, то соответствующая задача поиска экстремума является задачей линейного программирования. Если хотя бы одна из указанных функций нелинейна, то соответствующая задача поиска экстремума является задачей нелинейного программирования [26,28,40].

3.1.2. Классификация задач линейного программирования

Линейное программирование (ЛП) – один из первых и наиболее подробно изученных разделов математического программирования. Именно линейное программирование явилось тем разделом, с которого и начала развиваться сама дисциплина "математическое программирование". Термин "программирование" в названии дисциплины ничего общего с термином "программирование (т.е. составление программы) для ЭВМ" не имеет, т.к. дисциплина "линейное программирование" возникла еще до того времени, когда ЭВМ стали широко применяться для решения математических, инженерных, экономических и др. задач.

Термин "линейное программирование" возник в результате неточного перевода английского "linear programming". Одно из значений слова "programming" - составление планов, планирование. Следовательно, правильным переводом английского "linear programming" было бы не "линейное программирование", а "линейное планирование", что более точно отражает содержание дисциплины. Однако, термины линейное программирование, нелинейное программирование, математическое программирование и т.д. в нашей литературе стали общепринятыми и поэтому будут сохранены.

Итак, линейное программирование возникло после второй мировой войны и стало быстро развиваться, привлекая внимание математиков, экономистов и инженеров благодаря возможности широкого практического применения, а также математической стройности [26,28,40,94].

Можно сказать, что линейное программирование применимо для решения математических моделей тех процессов и систем, в основу которых может быть положена гипотеза линейного представления реального мира.

Линейное программирование применяется при решении экономических задач, в таких задачах как управление и планирование производства; в задачах определения оптимального размещения оборудования на морских судах, в цехах; в задачах определения оптимального плана перевозок груза (транспортная задача); в задачах оптимального распределения кадров и т.д.

Задача линейного программирования (ЛП), как уже ясно из сказанного выше, состоит в нахождении минимума (или максимума) линейной функции при линейных ограничениях.

Общая форма задачи имеет вид: найти $\min cx$ при условиях

$$a_i - b_i \geq 0, i \in I_1,$$
$$a_i - b_i = 0, i \in I_2,$$
$$x_j \geq 0, j \in J_1,$$

где

$$I_1 \cup I_2 = \{1,...,m\},\ I_1 \cap I_2 = \varnothing,\ J_1 \subset \{1,...n\},\ x = (x_1,...,x_n)^T,$$
$$c = (c_1...,c_n), a_i = (a_{i1},...,a_{in}),\ i = 1,...,m.$$

Здесь и далее нам удобнее считать c и a_i вектор - строками, а x и $b = (b_1,...,b_m)^T$ - вектор столбцами.

Наряду с общей формой широко используются также каноническая и стандартная формы. Как в канонической, так и в стандартной форме

$$J_1 = \{1,...,n\},$$

т.е. все переменные в любом допустимом решении задачи должны принимать неотрицательные значения (такие переменные принято называть неотрицательные в отличие от так называемых свободных переменных, на область значений которых подобное ограничение не накладывается). Отличие же между этими формами состоит в том, что в одном случае $I_2 = 0$, а в другом - $I_1 = 0$.

Задача ЛП в канонической форме:

$$w = cx \to \min,$$
$$Ax = b,$$
$$x \geq 0.$$

Задача ЛП в стандартной форме:

$$w = cx \to \min,$$
$$Ax \geq b,$$
$$x \geq 0.$$

В обоих случаях A есть матрица размерности $m \times n$, i-я строка которой совпадает с вектором a_i.

Задача ЛП в общей форме сводится (в определенном смысле) к задаче ЛП в канонической (стандартной) форме. Под этим понимается существование общего способа построения по исходной задаче (в общей форме) новой задачи ЛП (в нужной нам форме), любое оптимальное решение которой "легко" преобразуется в оптимальное решение исходной задачи и наоборот. (Фактически, связь между этими задачами оказывается еще более тесной). Тем самым мы получаем возможность, не теряя общности, заниматься изучением задач ЛП, представленных либо в канонической, либо в стандартной форме. Ввиду этого наши дальнейшие рассмотрения задач ЛП будут посвящены, главным образом, задачам в канонической форме.

Динамическое программирование служит эффективным методом решения задач оптимизации дискретных многостадийных процессов, для которых критерий оптимальности задается как аддитивная функция критериев оптимальности отдельных стадий. Без особых затруднений указанный метод можно распространить и на случай, когда критерий оптимальности задан в другой форме, однако при этом обычно увеличивается размерность отдельных стадий.

По существу метод динамического программирования представляет собой алгоритм определения оптимальной стратегии управления на всех стадиях процесса. При этом закон управления на каждой стадии находят путем решения частных задач оптимизации последовательно для всех стадий процесса с помощью методов исследования функций классического анализа или методов нелинейного программирования. Результаты решения обычно не могут быть выражены в аналитической форме, а получаются в виде таблиц [26,28,40].

Ограничения на переменные задачи не оказывают влияния на общий алгоритм решения, а учитываются при решении частных задач

оптимизации на каждой стадии процесса. При наличии ограничений типа равенств иногда даже удается снизить размерность этих частных задач за счет использования множителей Лагранжа. Применение метода динамического программирования для оптимизации процессов с распределенными параметрами или в задачах динамической оптимизации приводит к решению дифференциальных уравнений в частных производных. Вместо решения таких уравнений зачастую значительно проще представить непрерывный процесс как дискретный с достаточно большим числом стадий. Подобный прием оправдан особенно в тех случаях, когда имеются ограничения на переменные задачи и прямое решение дифференциальных уравнений осложняется необходимостью учета указанных ограничений.

При решении задач методом динамического программирования, как правило, используют вычислительные машины, обладающие достаточным объемом памяти для хранения промежуточных результатов решения, которые обычно получаются в табличной форме.

Принцип максимума применяют для решения задач оптимизации процессов, описываемых системами дифференциальных уравнений. Достоинством математического аппарата принципа максимума является то, что решение может определяться в виде разрывных функций; это свойственно многим задачам оптимизации, например задачам оптимального управления объектами, описываемыми линейными дифференциальными уравнениями.

Нахождение оптимального решения при использовании принципа максимума сводится к задаче интегрирования системы дифференциальных уравнений процесса и сопряженной системы для вспомогательных функций при граничных условиях, заданных на обоих концах интервала интегрирования, т. е. к решению краевой задачи. На область изменения переменных могут быть наложены ограничения. Систему дифференциальных уравнений интегрируют, применяя обычные программы на цифровых вычислительных машинах.

Принцип максимума для процессов, описываемых дифференциальными уравнениями, при некоторых предположениях является достаточным условием оптимальности. Поэтому дополнительной проверки на оптимум получаемых решений обычно не требуется [26,28,40].

Для дискретных процессов принцип максимума в той же формулировке, что и для непрерывных, вообще говоря, несправедлив. Однако условия оптимальности, получаемые при его применении для многостадийных процессов, позволяют найти достаточно удобные алгоритмы оптимизации.

Методы нелинейного программирования применяют для решения оптимальных задач с нелинейными функциями цели. На независимые переменные могут быть наложены ограничения также в виде нелинейных соотношений, имеющих вид равенств или неравенств. По существу методы нелинейного программирования используют, если ни один из перечисленных выше методов не позволяет сколько-нибудь продвинуться в решении оптимальной задачи. Поэтому указанные методы иногда называют также прямыми методами решения оптимальных задач.

Для получения численных результатов важное место отводится нелинейному программированию и в решении оптимальных задач такими методами, как динамическое программирование, принцип максимума и т. п. на определенных этапах их применения.

Названием “методы нелинейного программирования” объединяется большая группа численных методов, многие из которых приспособлены для решения оптимальных задач соответствующего класса. Выбор того или иного метода обусловлен сложностью вычисления критерия оптимальности и сложностью ограничивающих условий, необходимой точностью решения, мощностью имеющейся вычислительной машины и т.д. Ряд методов нелинейного программирования практически постоянно используется в сочетании с другими методами оптимизации, как, например, метод сканирования в динамическом программировании. Кроме того, эти методы служат основой построения систем автоматической оптимизации - оптимизаторов, непосредственно применяющихся для управления производственными процессами [26,28,94,121].

Геометрическое программирование есть метод решения одного специального класса задач нелинейного программирования, в которых критерий оптимальности и ограничения задаются в виде полиномов - выражений, представляющих собой сумму произведений степенных функций от независимых переменных. С подобными задачами иногда приходится сталкиваться в проектировании. Кроме того, некоторые задачи нелинейного программирования иногда можно свести к указанному представлению, используя

аппроксимационное представление для целевых функций и ограничений.

Специфической особенностью методов решения оптимальных задач (за исключением методов нелинейного программирования) является то, что до некоторого этапа оптимальную задачу решают аналитически, т. е. находят определенные аналитические выражения, например, системы конечных или дифференциальных уравнений, откуда уже отыскивают оптимальное решение. В отличие от указанных методов при использовании методов нелинейного программирования, которые, как уже отмечалось выше, могут быть названы прямыми, применяют информацию, получаемую при вычислении критерия оптимальности, изменение которого служит оценкой эффективности того или иного действия.

Важной характеристикой любой оптимальной задачи является ее размерность *n,* равная числу переменных, задание значений которых необходимо для однозначного определения состояния оптимизируемого объекта. Как правило, решение задач высокой размерности связано с необходимостью выполнения большого объема вычислений. Ряд методов (например, динамическое программирование и дискретный принцип максимума) специально предназначен для решения задач оптимизации процессов высокой размерности, которые могут быть представлены как многостадийные процессы с относительно невысокой размерностью каждой стадии [28,40].

В большинстве инженерных задач построение математической модели не удается свести к задаче линейного программирования.

Математические модели в задачах проектирования реальных объектов или технологических процессов должны отражать реальные протекающие в них физические и, как правило, нелинейные процессы. Переменные этих объектов или процессов связанны между собой физическими нелинейными законами, такими, как законы сохранения массы или энергии. Они ограничены предельными диапазонами, обеспечивающими физическую реализуемость данного объекта или процесса. В результате, большинство задач математического программирования, которые встречаются в научно-исследовательских проектах и в задачах проектирования – это задачи нелинейного программирования (НП).

Пусть в математической модели проектируемого объекта или процесса непрерывная функция $F(\overline{X})$ - представляет собой функцию цели (функцию качества),

$$h_1(\overline{X}), h_2(\overline{X}), h_3(\overline{X}),..., h_m(\overline{X}), \quad i=\overline{1,m} \quad -$$

задают ограничения в виде равенств

$$g_{m+1}(\overline{X}), g_{m+2}(\overline{X}), g_{m+3}(\overline{X}),..., g_p(\overline{X}), \quad j=\overline{m+1,p}, \quad -$$

задают ограничения в виде неравенств, где $\overline{X}=[x_1, x_2, x_3,..., x_n]$, $\overline{X} \in E^n$ - вектор параметров проектируемого объекта, процесса или системы, оптимальные значения которых должны быть найдены.

Тогда задача нелинейного программирования может быть сформулирована следующим образом:

найти вектор $\overline{X}=[x_1, x_2, x_3,..., x_n]$, $\overline{X} \in E^n$, доставляющий минимум (максимум) целевой функции $F(\overline{X})$ при m линейных и (или) нелинейных ограничений в виде равенств

$$h_i(\overline{X})=0, \quad i=\overline{1,m}$$

и (p-m) линейных и (или) нелинейных ограничений в виде неравенств

$$g_j(\overline{X})>0, \quad j=\overline{m+1,\text{p}}.$$

В течение последних двух десятилетий из нелинейного программирования выделились самостоятельные разделы [26,28,40]:

- выпуклое программирование,
- квадратичное программирование,
- целочисленное программирование,
- стохастическое программирование,
- динамическое программирование и др.

Задачи выпуклого программирования – это задачи, в которых определяется минимум выпуклой функции (или максимум вогнутой), заданной на выпуклом замкнутом множестве. Эти задачи среди задач нелинейного программирования наиболее изучены.

Среди задач выпуклого программирования более подробно изучены задачи квадратичного программирования. В этих задачах целевая функция – квадратична, а ограничения – линейны.

В задачах целочисленного программирования неизвестные параметры могут принимать только целочисленные значения.

В задачах стохастического программирования в целевой функции или в функциях ограничений содержатся случайные величины, которые подчиняются законам теории вероятностей.

В задачах динамического программирования ограничения содержат как параметр время и при этом описываются

дифференциальными уравнениями. Процесс нахождения решений в задачах динамического программирования является многоэтапным.

3.1.3. Классификация методов нелинейного программирования

Для решения задачи нелинейного программирования было предложено много методов, которые можно классифицировать по различным признакам.

По количеству локальных критериев в целевой функции методы нелинейного программирования делятся на:

- однокритериальные,
- многокритериальные.

По длине вектора $\overline{x}$ методы делятся на:

- однопараметрические или одномерные (n=1),
- многопараметрические или многомерные (n>1).

По наличию ограничений методы нелинейного программирования делятся на:

- без ограничений (безусловная оптимизация),
- с ограничениями (условная оптимизация).

По типу информации, используемой в алгоритме поиска экстремума методы делятся на:

- методы прямого поиска, т.е. методы, в которых при поиске экстремума целевой функции используются только ее значения;
- градиентные методы первого порядка, в которых при поиске экстремума функции используются значения ее первых производных;
- градиентные методы второго порядка, в которых при поиске экстремума функции наряду с первыми производными используются и вторые производные.

Ни один метод нелинейного программирования не является универсальным. В каждом конкретном случае необходимо приспосабливать применяемый метод к особенностям решаемой задачи [26,28,40,94].

3.1.4. Классический метод определения условного экстремума

Задача нелинейного программирования (задача НП) в общем виде формулируется так:

$$\textit{максимизировать } f(x_1, x_2, \ldots x_n)$$

при ограничениях

$$g_1(x_1, x_2, ..., x_n) \geq 0;$$
$$g_2(x_1, x_2, ..., x_n) \geq 0;$$
$$.....$$
$$g_m(x_1, x_2, ..., x_n) \geq 0;$$

где функции $f(x_1, x_2, ...x_n), g_1(x_1, x_2, ...x_n) \geq 0,\ i = \overline{1,m}$ нелинейны.

В отличие от задачи ЛП для задач НП нет универсального метода решения.

В задаче ЛП допустимое множество R всегда является выпуклым с конечным числом крайних точек. Поэтому воспользовавшись симплекс-методом и перебрав только крайние точки, можно за конечное число шагов найти оптимальное решение. В задачах НП, наоборот, выпуклость допустимого множества и конечность числа его крайних точек совсем необязательны. Это и служит причиной основной трудности решения задач НП [26,28,40].

3.1.5. Метод полного перебора (метод сеток)

Многомерные задачи, естественно, являются более сложными и трудоемкими, чем одномерные, причем обычно трудности при их решении возрастают при увеличении размерности. Для того чтобы вы лучше почувствовали это, возьмем самый простой по своей идее приближенный метод поиска наименьшего значения функции. Покроем рассматриваемую область сеткой G с шагом h (рис. 3.1) и определим значения функции в ее узлах. Сравнивая полученные числа между собой, найдем среди них наименьшее и примем его приближенно за наименьшее значение функции для всей области.

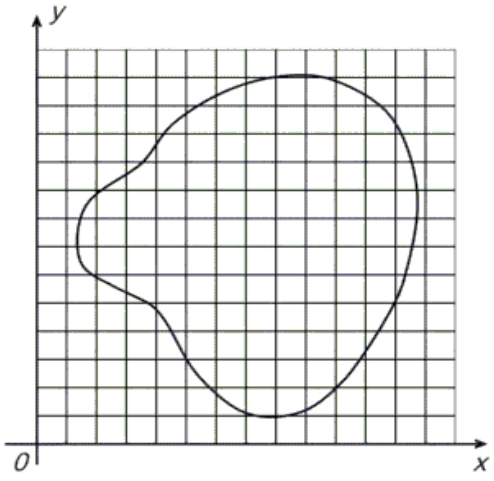


Рис. 3.1. Покрытие рассматриваемой области сеткой

Как мы уже говорили выше, данный метод используется для решения одномерных задач. Иногда он применяется также для решения двумерных, реже трехмерных задач. Однако для задач большей размерности он практически непригоден из-за слишком большого времени, необходимого для проведения расчетов. Действительно, предположим, что целевая функция зависит от пяти переменных, а область определения G является пятимерным кубом, каждую сторону которого при построении сетки мы делим на 40 частей. Тогда общее число узлов сетки будет равно $41^5 \approx 10^8$. Пусть вычисление значения функции в одной точке требует 1000 арифметических операций (это немного для функции пяти переменных). В таком случае общее число операций составит 10^{11}. Если в нашем распоряжении имеется ЭВМ с быстродействием 1 млн. операций в секунду, то для решения задачи с помощью данного метода потребуется 10^5 секунд, что превышает сутки непрерывной работы. Добавление еще одной независимой переменной увеличит это время в 40 раз. Проведенная оценка показывает, что для больших задач оптимизации метод сплошного перебора непригоден. Иногда сплошной перебор заменяют случайным поиском [3,26,28,40]. В этом случае точки сетки просматриваются не подряд, а в случайном порядке. В результате поиск наименьшего значения целевой функции существенно ускоряется, но теряет свою надежность.

3.1.6. Метод покоординатного спуска

Рассмотрим функцию двух переменных. Ее линии постоянного уровня представлены на рис. 3.2, а минимум лежит в точке (x_1^*, x_2^*). (Напомним, что линией постоянного уровня называется кривая в двумерном сечении пространства параметров (в данном случае в плоскости (x_1, x_2), значение функции на которой - константа). Простейшим методом поиска является метод покоординатного спуска. Из точки A мы производим поиск минимума вдоль направления оси x_1 и, таким образом, находим точку B, в которой касательная к линии постоянного уровня параллельна оси x_1. Затем, производя поиск из точки B в направлении оси x_2, получаем точку C, производя поиск параллельно оси x_1, получаем точку D, и т.д. Таким образом, мы приходим к оптимальной точке. Любой из одномерных методов, описанных в предыдущей главе, может быть использован здесь для поиска вдоль оси. Очевидным образом эту идею можно применить для функций n переменных.

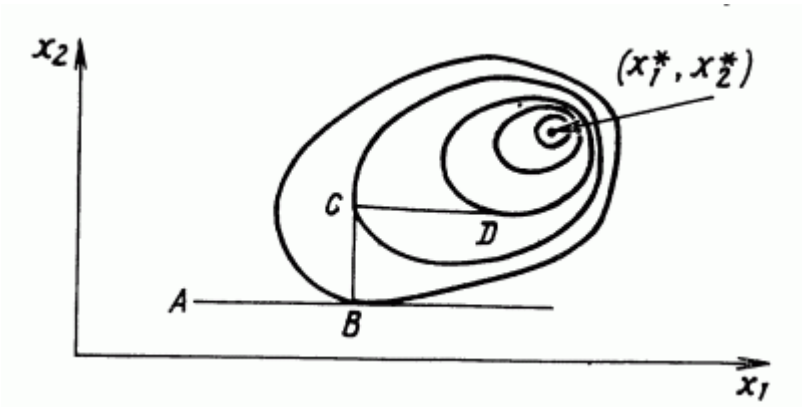


Рис. 3.2. Линии постоянного уровня

Рассмотрим данный метод более детально на примере некоторой целевой функции.

Пусть нужно найти наименьшее значение целевой функции $u = f(M) = f(x_1, x_2, ...x_n)$. Здесь через M обозначена точка n-мерного пространства с координатами $x_1, x_2, ..., x_n : M = (x_1, x_2, ..., x_n)$. Выберем какую-нибудь начальную точку $M_0 = (x_1^0, x_2^0, ..., x_n^0)$ и рассмотрим функцию f при фиксированных значениях всех переменных, кроме первой: $f(x_1^0, x_2^0, ..., x_n^0)$. Тогда она превратится в функцию одной переменной x_1. Изменяя эту переменную, будем двигаться от начальной точки $x_1 = x_1^0$ в сторону убывания функции, пока не дойдем до ее минимума при $x_1 = x_1^1$, после которого она начинает возрастать. Точку с координатами $f(x_1^1, x_2^0, ..., x_n^0)$ обозначим через M^1, при этом $f(M^0) \geq f(M^1)$.

Фиксируем теперь переменные: $x_1 = x_1^1, x_3 = x_3^0, ..., x_n = x_n^0$ и рассмотрим функцию f как функцию одной переменной $x_2 : f(x_1^1, x_2, x_3^0, ..., x_n^0)$. Изменяя x_2, будем опять двигаться от начального значения $x_2 = x_2^0$ в сторону убывания функции, пока не дойдем до минимума при $x_2 = x_2^1$. Точку с координатами $(x_1^1, x_2, x_3^0, ..., x_n^0)$ обозначим через M^2, при этом $f(M^1) \geq f(M^2)$. Проведем такую же минимизацию целевой функции по переменным $x_3, x_4, ..., x_n$. Дойдя до переменной x_n, снова вернемся к x_1 и продолжим процесс.

Эта процедура вполне оправдывает название метода. С ее помощью мы построим последовательность точек $M^0, M^1, M^2, ...$ которой соответствует монотонная последовательность значений функции $f(M^0) \geq f(M^1) \geq f(M^2)....$. Обрывая ее на некотором шаге k,

можно приближенно принять значение функции $f(M^k)$ за ее наименьшее значение в рассматриваемой области (рис. 3.3).

Отметим, что данный метод сводит задачу поиска наименьшего значения функции нескольких переменных к многократному решению одномерных задач оптимизации. Если целевая функция задана явной формулой и является дифференцируемой, то мы можем вычислить ее частные производные и использовать их для определения направления убывания функции по каждой переменной и поиска соответствующих одномерных минимумов [26,28,40].

На рис. 3.3. изображены линии уровня некоторой функции двух переменных. $u = f(x, y)$ Вдоль этих линий функция сохраняет постоянные значения, равные 1, 3, 5, 7, 9. Показана траектория поиска ее наименьшего значения, которое достигается в точке *O*, с помощью метода покоординатного спуска.

При этом нужно ясно понимать, что рисунок служит только для иллюстрации метода. Когда мы приступаем к решению реальной задачи оптимизации, такого рисунка, содержащего в себе готовый ответ, у нас, конечно, нет.

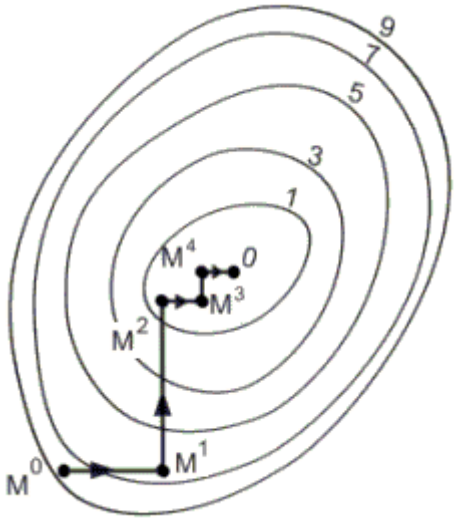


Рис. 3.3. Линии уровня функции двух переменных

Теоретически данный метод эффективен в случае единственного минимума функции. Но на практике он оказывается слишком медленным. Поэтому были разработаны более сложные методы, использующие больше информации на основании уже полученных значений функции. Было предложено несколько функций, которые

из-за своих свойств являются тестовыми для таких методов. Ниже приведено несколько примеров таких функций [169].

Функция Розенброка:

$$f(x_1, x_2) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2; \quad x^* = (1;1).$$

Функция Пауэлла:

$$f(x) = (x_1 - 10x_2)^2 + 5(x_3 - x_4)^2 + (x_2 - 2x_3)^4 + 10(x_1 - x_4)^4; \quad x^* = (0;0;0;0).$$

Двумерная экспоненциальная функция:

$$f(\mathrm{x}_1, \mathrm{x}_2) = \sum_a \left[\left(e^{-ax_1} - e^{-ax_2} \right) - (\mathrm{e}^{-a} - \mathrm{e}^{-10a}) \right]^2, \ \textit{где } a = 0,1(0,1)1^*; \ x^* = (1;10).$$

3.1.7. Метод градиентного спуска

Рассмотрим функцию f считая для определенности, что она зависит от трех переменных x, y, z. Вычислим ее частные производные $\frac{df}{dx}, \frac{df}{dy}, \frac{df}{dz}$ и образуем с их помощью вектора, который называют градиентом функции:

$$grad\ f(x,y,z) = \frac{df}{dx}(x,y,z)i + \frac{df}{dy}(x,y,z)\,j + \frac{df}{dz}(x,y,z)k.$$

Здесь i, j, k - единичные векторы, параллельные координатным осям. Частные производные характеризуют изменение функции по каждой независимой переменной в отдельности. Образованный с их помощью вектор градиента дает общее представление о поведении функции в окрестности точки (x, y, z). Направление этого вектора является направлением наиболее быстрого возрастания функции в данной точке. Противоположное ему направление, которое часто называют антиградиентным, представляет собой направление наиболее быстрого убывания функции. Модуль градиента

$$|grad\ f(x,y,z)| = \sqrt{\left(\frac{df}{dx}(x,y,z)i\right)^2 + \left(\frac{df}{dy}(x,y,z)\,j\right)^2 + \left(\frac{df}{dz}(x,y,z)k\right)^2}$$

определяет скорость возрастания и убывания функции в направлении градиента и антиградиента [28,40]. Для всех остальных направлений скорость изменения функции в точке (x, y, z) меньше модуля градиента. При переходе от одной точки к другой как направление градиента, так и его модуль, вообще говоря, меняются. Понятие градиента естественным образом переносится на функции любого числа переменных.

Перейдем к описанию метода градиентного спуска. Основная его идея состоит в том, чтобы двигаться к минимуму в направлении

наиболее быстрого убывания функции, которое определяется антиградиентом. Эта идея реализуется следующим образом.

Выберем каким-либо способом начальную точку, вычислим в ней градиент рассматриваемой функции и сделаем небольшой шаг в обратном, антиградиентном направлении. В результате мы придем в точку, в которой значение функции будет меньше первоначального. В новой точке повторим процедуру: снова вычислим градиент функции и сделаем шаг в обратном направлении. Продолжая этот процесс, мы будем двигаться в сторону убывания функции. Специальный выбор направления движения на каждом шаге позволяет надеяться на то, что в данном случае приближение к наименьшему значению функции будет более быстрым, чем в методе покоординатного спуска.

Метод требует вычисления градиента целевой функции на каждом шаге. Если она задана аналитически, то это, как правило, не проблема: для частных производных, определяющих градиент, можно получить явные формулы. В противном случае частные производные в нужных точках приходится вычислять приближенно, заменяя их соответствующими разностными отношениями:

$$\frac{df}{dx} \approx \frac{f(x_1,...,x_i+\Delta x_i,...x_n)-f(x_1,...,x_i,...,x_n)}{\Delta x_i}.$$

Отметим, что при таких расчетах Δx_i, нельзя брать слишком малым, а значения функции нужно вычислять с достаточно высокой степенью точности, иначе при вычислении разности

$$\Delta f = f(x_1,...,x_i+\Delta x_i,...x_n)-f(x_1,...,x_i,...,x_n).$$

3.1.8. Метод наискорейшего спуска

Суть данного метода заключается в том, что с помощью упомянутого ранее метода покоординатного спуска осуществляется поиск из заданной точки в направлении, параллельном одной из осей, до точки минимума в данном направлении [26,28,40]. Затем поиск производится в направлении, параллельном другой оси, и т.д. Направления, конечно, фиксированы. Кажется разумным попытаться модифицировать этот метод таким образом, чтобы на каждом этапе поиск точки минимума производился вдоль "наилучшего" направления. Не ясно, какое направление является "наилучшим", но известно, что направление градиента является направлением наискорейшего возрастания функции. Следовательно, противоположное направление является направлением

наискорейшего убывания функции. Это свойство может быть обосновано следующим образом.

Предположим, что осуществляется перемещение из точки x в следующую точку $x + hd$, где d - некоторое направление, а h - шаг некоторой длины. Следовательно, перемещение производится из точки $(x_1, x_2, ..., x_n)$ в точку $(x_1 + \mathrm{zx}_1, x_2 + \mathrm{zx}_2, ..., x_n + zx_n)$, где

$$zx_i = hd_i \tag{3.1}$$

а d_j - косинусы направления d, такие, что

$$\sum_{i=1}^{n} d_i^2 = 1. \tag{3.2}$$

Изменение значений функции определяется соотношениями

$$\begin{aligned} df &= f(x_1 + \delta x_1, x_2 + \delta x_2, ..., x_n + \delta x_n) - f(x_1, x_2, ..., x_n) = \\ &= \frac{\partial f}{\partial x_1}\delta x_1 + \frac{\partial f}{\partial x_2}\delta x_2 + ... + \frac{\partial f}{\partial x_n}\delta x_n \end{aligned} \tag{3.3}$$

с точностью до первого порядка zx_i, причем частные производные вычисляются в точке x. Как следует выбрать направления d_i, удовлетворяющие уравнению (3.2), чтобы получить наибольшее значение изменения функции df? Здесь возникает задача максимизации с ограничением. Применим метод множителей Лагранжа, с помощью которого определим функцию

$$\varphi(d_1, d_2, ..., d_n) = df + \lambda\left(\sum d_1^2 - 1\right). \tag{3.4}$$

Величина df, удовлетворяющая ограничению (3.2), достигает максимума, когда функция

$$\varphi(d_1, d_2, ..., d_n) = h\left(\frac{\partial f}{\partial x_1}d_1 + \frac{\partial f}{\partial x_2}d_2 + ... + \frac{\partial f}{\partial x_n}d_n\right) + \lambda(d_1^2 + d_2^2 + ... + d_n^2 - 1)$$

достигает максимума. Ее производная

$$\frac{\partial \varphi}{\partial d_j} = h\frac{\partial \varphi}{\partial x_j} + \lambda d_j, \ \ при \ \ j = 1, 2, ..., n.$$

Если $\frac{\partial \varphi}{\partial d_j} = 0$, то

$$d_j = -\frac{h}{2\lambda}\frac{\partial f}{\partial x_j}. \tag{3.5}$$

Следовательно,

$$\frac{d_1}{\dfrac{\partial f}{\partial x_1}} = \frac{d_2}{\dfrac{\partial f}{\partial x_2}} = ... = \frac{d_n}{\dfrac{\partial f}{\partial x_n}}. \tag{3.6}$$

Тогда $d_i \approx \dfrac{\partial f}{\partial x_i}$ и направление d параллельно направлению $\nabla f(x)$ в точке x.

Таким образом, наибольшее локальное возрастание функции для заданного малого шага h имеет место, когда d есть направление $\nabla f(x)$ или $g(x)$. Поэтому направлением наискорейшего спуска является направление

$$\nabla f(x) \text{ или } g(x). \qquad (3.7)$$

В более простом виде уравнение (3.3) можно записать так:

$$df - |\nabla f(x)||dx|\cos\theta,$$

где θ - угол между векторами $\nabla f(x)$ и dx. Для заданной величины dx мы минимизируем df, выбирая $\theta = 180^0$, чтобы направление dx совпадало с направлением - $\nabla f(x)$.

Направление градиента перпендикулярно в любой точке линии постоянного уровня, поскольку вдоль этой линии функция постоянна. Таким образом, если $(d_1, d_2, ..., d_n)$ - малый шаг вдоль линии уровня, то (рис. 3.4)

$$f(x_1 + d_1, x_2 + d_2, ..., x_n + d_n) = f(x_1, x_2, ..., x_n)$$

и, следовательно,

$$df = \sum_{j=1}^{n} \frac{\partial f}{\partial x_j} d_j = [\nabla f(x)]^T d = 0. \qquad (3.8)$$

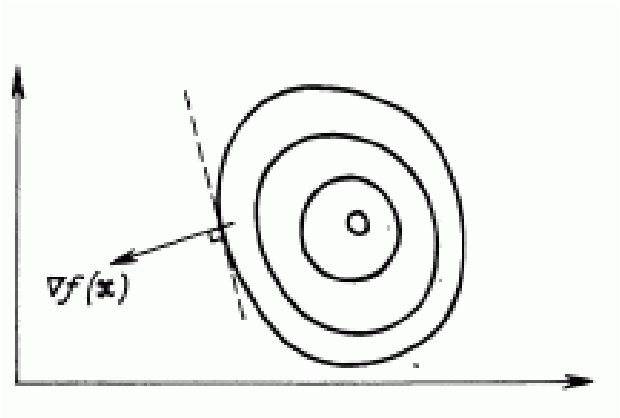


Рис. 3.4. Линии уровня

В методе наискорейшего спуска желательно использовать рассмотренное свойство направления градиента. Поэтому, если мы находимся в точке x_i на некотором шаге процесса оптимизации, то поиск минимума функции осуществляется вдоль направления - $\nabla f(x_j)$.

Данный метод является итерационным. На шаге i точка минимума аппроксимируется точкой x_i. Следующей аппроксимацией является точка

$$x_{i+1} = x_i - \lambda_i \nabla f(x_i), \tag{3.9}$$

где λ_i - значение λ, минимизирующее функцию

$$\varphi(\lambda) = f\left[x_i - \lambda \nabla f(x_i)\right]. \tag{3.10}$$

Значение λ_i может быть найдено с помощью одного из методов одномерного поиска. Блок-схема метода наискорейшего спуска приведена на рис. 3.5.

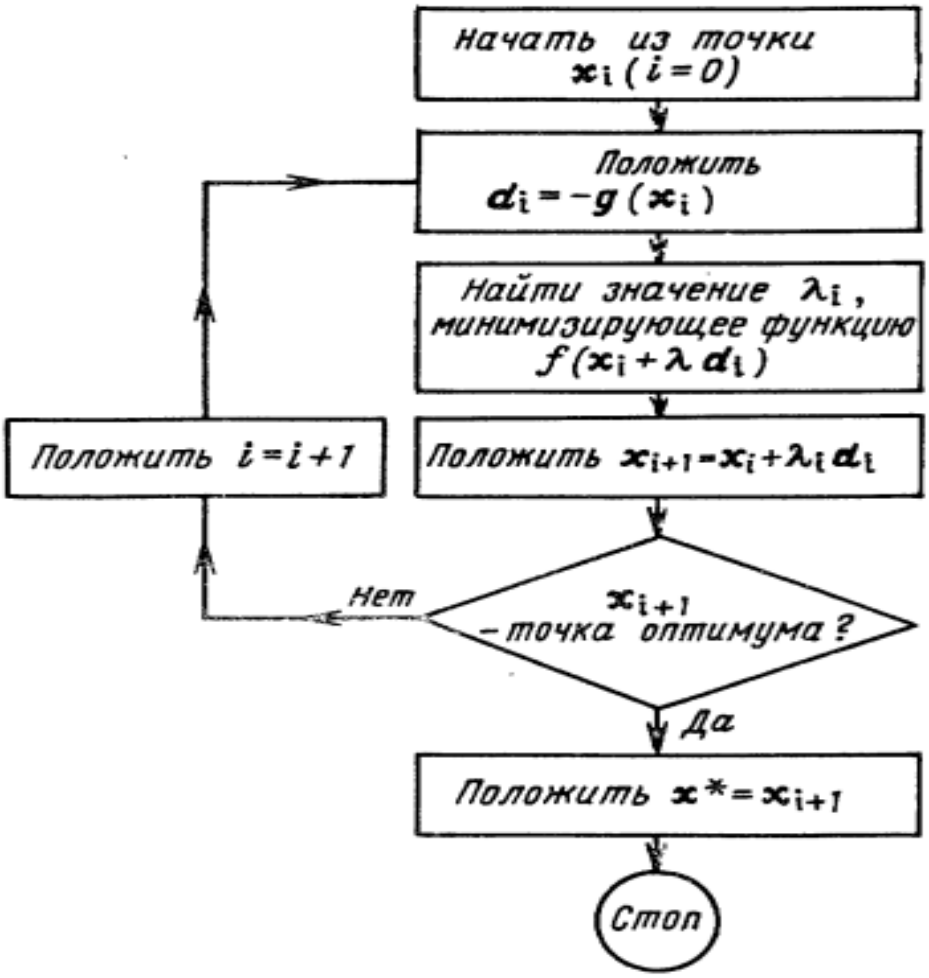


Рис. 3.5. Блок-схема метода наискорейшего спуска

Выше нами были рассмотрены 3 варианта методов спуска, такие, как метод покоординатного спуска, метод градиентного спуска и метод наискорейшего спуска, и при этом наглядно показали, как хорошо они работают. В результате у вас могло сложиться

впечатление, что проблема решена. На самом деле это не так. Все было хорошо потому, что был выбран "удобный" пример. Но посмотрите на рис. 3.4. На нем также показаны линии уровня некоторой функции. Линии уровня сильно вытянуты в одном направлении и сплющены в другом. Они напоминают рельеф местности с оврагом. Этот случай крайне неудобен для описанных выше методов.

Действительно, попытаемся найти наименьшее значение такой функции с помощью градиентного спуска. Двигаясь все время в направлении антиградиента, мы быстро спустимся на дно "оврага" и, поскольку движение идет хотя и маленькими, но конечными шагами, проскочим его. Оказавшись на противоположной стороне "оврага" и вычислив там градиент функции, мы будем вынуждены развернуться почти на 180° и сделать один или несколько шагов в обратном направлении. При этом мы снова проскочим дно "оврага" и вернемся на его первоначальную сторону. Продолжая этот процесс, мы вместо того, чтобы двигаться по дну "оврага" в сторону его понижения, будем совершать зигзагообразные скачки поперек "оврага", почти не приближаясь к цели. Таким образом, в случае "оврага" (этот нематематический термин прочно закрепился в литературе) описанные выше методы спуска оказываются неэффективными.

Для борьбы с "оврагами" был предложен ряд специальных приемов. Один из них заключается в следующем. Из двух близких точек совершают градиентный спуск на дно "оврага". Потом соединяют найденные точки прямой и делают вдоль нее большой (овражный) шаг. Из найденной точки снова спускаются на дно "оврага" и делают второй овражный шаг. В результате, двигаясь достаточно быстро вдоль "оврага", приближаемся к искомому наименьшему значению целевой функции (см. рис. 3.6). Такой метод достаточно эффективен для функций двух переменных, однако при большем числе переменных могут возникнуть трудности.

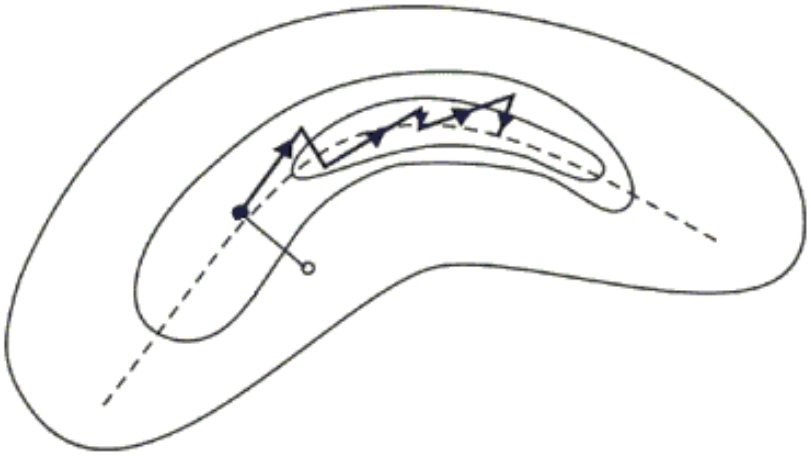


Рис. 3.6. Поиск наименьшего значения функции в случае "оврага"

Все описанные выше методы приспособлены к случаю, когда наименьшее значение функции достигается внутри рассматриваемой области, и становятся малоэффективными, если наименьшее значение достигается на границе или вблизи нее. Для решения таких задач приходится разрабатывать специальные методы. Мы не будем на них останавливаться. Вам должно быть и без того ясно, что большое число специальных методов - это признак слабости, а не силы. Ведь приступая к решению практической задачи, мы, как правило, не знаем всех ее особенностей и не можем сразу выбрать наиболее эффективный метод.

На рис. 3.7 приведены линии уровня функции с двумя локальными минимумами в точках O_1 и O_2. Такие функции принято называть многоэкстремальными. Сравнивая между собой значения функции в точках O_1 и O_2 $f_1=3$, $f_2=1$, находим, что наименьшее значение функция достигает в точке O_2.

Представьте себе теперь, что, не имея перед глазами рис. 3.7 и не зная о многоэкстремальности функции, мы начали поиск наименьшего значения с помощью метода градиентного спуска из точки A_1. Поиск приведет нас в точку O_1, которую ошибочно можно принять за искомый ответ. С другой стороны, если мы начнем поиск с точки A_2, то окажемся на правильном пути и быстро придем в точку O_2.

Как бороться с многоэкстремальностью? Универсального ответа на этот вопрос нет. Самый простой прием состоит в том, что проводят поиск несколько раз, начиная его с разных точек. Если при этом получаются разные ответы, то сравнивают в них значения целевой функции и выбирают наименьшее. Расчеты останавливают

после того, как несколько новых поисков не меняют полученного ранее результата. Выбор начальных точек поиска, обоснованность прекращения расчетов в значительной степени зависят от опыта и интуиции специалистов, решающих задачу.

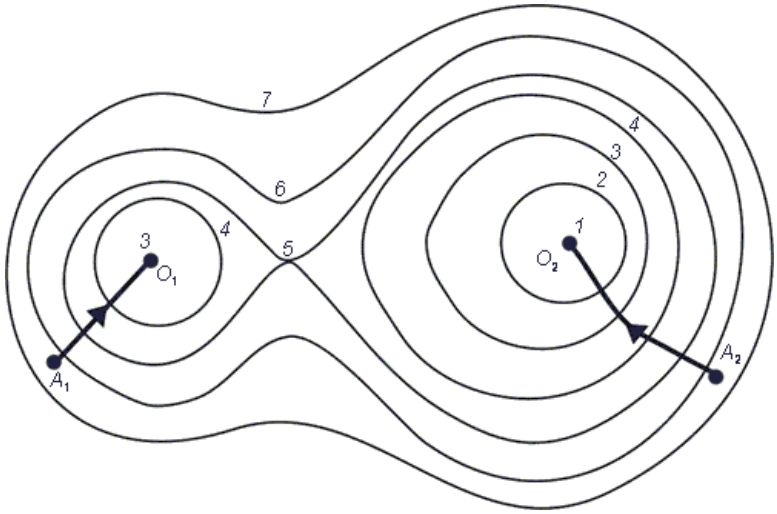


Рис. 3.7. Пример функции с двумя локальными минимумами в точках *O1* и *O2*

3.2. Анализ решения задач с помощью метода случайного поиска

Проблема оптимизации в настоящее время возникает практически во всех областях науки и человеческой деятельности и привлекает особое внимание специалистов всех областей.

Известно, что для различных классов экстремальных задач наиболее пригодны различные методы глобального поиска. Существующие в настоящее время методы можно разбить на три большие группы:

- детерминированные,
- случайные,
- комбинированные.

Случайный поиск (СП) имеет большую эффективность и значительно превосходит обычные методы поиска в случае дискретно-непрерывной оптимизации, не требует дополнительного исследования функции и применяется в случае большого количества параметров. Следует помнить, что в подобных алгоритмах нахождение точного минимума не требуется – решением может считаться любое значение, которое лучше некоторой заданной величины [3,34].

Случайность может проявляться в следующих вещах:

1. моделируется закон распределения направления спуска,
2. закон распределения длины шага спуска,
3. координаты вектора $x = (x_1,...,x_n)$,
4. размер окрестности поиска; и т.д.

В данной работе рассматривается алгоритм, относящийся к семейству адаптивных методов и характеризующийся тем, что в процессе работы накапливает информацию о целевой функции и использует ее для увеличения вероятности сходимости к оптимуму [3].

Характеристики эффективного поиска

Эффективность процесса поиска определяется его характеристиками, которые получаются на основе выбранных критериев. Эти характеристики описывают основные свойства поиска как процесса оптимизации.

Будем разделять все характеристики поиска на два класса: класс локальных и класс нелокальных, интегральных характеристик.

Первый класс образуют характеристики, определяющие эффективность поиска на одном этапе (под этапом поиска подразумевается однократный процесс сбора информации и принятия решения, т.е. серия поисковых шагов с одним рабочим шагом). Второй класс значительно шире; он образуется характеристиками, определяющими эффективность всего процесса поиска от начала до конца оптимизации.

1) Локальные характеристики поиска:

- потери на поиск (средняя локальная скорость оптимизации, т.е. быстродействие поиска на одном этапе);
- вероятность ошибки (вероятность появления в процессе поиска ошибочного рабочего шага, т.е. если значение функции в новом состоянии превышает значение в исходном состоянии).

- Нелокальные характеристики:

- критерий точности решения;
- критерий числа шагов, необходимых для решения задачи с заданной точностью; естественно считать наилучшим тот алгоритм, который обеспечивает решение поставленной задачи при наименьшем числе шагов поиска и средней точностью не менее заданной;

- критерий надежности поиска (характеристикой надежности является вероятность оптимизации системы с заданной точностью решения); и др.

1. Общий вид алгоритма глобального СП

Все алгоритмы глобального СП могут быть записаны в виде следующего алгоритма [3]:

- Выбираем некоторое распределение вероятностей $P_0(dx)$ на B (где B - σ -алгебра борелевских подмножеств множества X), полагаем $k = 0$
- Моделируем некоторое число N_k раз распределение $P_k(dx)$, получаем точки $x_1^{(k)},...,x_{N_k}^{(k)}$, в которых вычисляем (возможно, со случайной ошибкой) значения Φ.
- Согласно определенному правилу конструируем распределение вероятностей $P_{k+1}(dx)$ на B.
- Проверяем условие останова, и если останова нет, то переходим к шагу 2 с заменой k на $k + 1$.

2. Задача поиска глобального решения

Пусть X — компактное метрическое пространство (множество оптимизации), Φ - ограниченная снизу функция, заданная на X (целевая функция). Задача поиска минимума функции Φ состоит в построении последовательности точек $x^{(1)}, x^{(2)},...,$ из X сходящейся в одной из точек $x^* = \arg\min \Phi$ глобального минимума функции Φ. Типы сходимости могут быть различными: от сходимости в метрике пространства X до сходимости с некоторой вероятностью.

Предположение о замкнутости X вместе с обязательным предположением о непрерывности целевой функции в окрестности точки x^* гарантирует то, что $x^* \in X$, т. е. глобальный максимум Φ в X достигается.

Таким образом, пусть требуется определить такой вектор

$$x^* = (x_1^*, x_2^*,..., x_n^*)^T, \qquad 0 \le x_i \le 1, \qquad i \in 1:n,$$

при котором целевая функция $\Phi(x^*)$ принимает минимальное значение. Будем считать, что дополнительные ограничения на переменные $x_1, x_2,..., x_n$ учтены при построении целевой функции (например, при помощи штрафных функций). Запишем поставленную задачу в общем виде:

$$\Phi(x) \to \min_{x \in X}, \tag{3.11}$$

где $x^* = (x_1^*, x_2^*, ..., x_n^*)^T \in [0,1]^n$.*

1. Постановка задачи

Будем рассматривать следующие задачи:

- Статистическое исследование алгоритма глобальной оптимизации и его модернизации на базе логистической кривой, сравнение с показательным законом.
- Исследование результатов в зависимости от параметров логистической кривой.
- Выявление наилучшего параметра (множества наилучших параметров), универсального для всех классов функций с помощью методов принятия решений.

- Реализация алгоритма поиска

Весь поиск разбивается на задаваемое пользователем число шагов n_{step}. На каждом шаге по определенному закону случайным образом выбираются значения вектора параметров x^k (k – номер шага), и подсчитывается значение целевой функции $\Phi^k = \Phi(x^k)$. Далее по формуле

$$\Phi_{\min}^k = \min\left\{\Phi^k, \Phi_{\min}^{k-1}\right\} \qquad (3.12)$$

определяется наименьшее значение, полученное за k шагов процесса поиска. После каждого расчета по формуле (3.12) закон, по которому выбираются значения переменных $x_i (i \in [1:n])$, изменяется таким образом, чтобы вероятность попадания в Δ - окрестность глобального минимума, задаваемую пользователем, исходя из требуемой точности решения задачи, увеличилась бы. Для этого используется информация, полученная на предыдущих шагах поиска.

Назовем интервал, на котором полученным на предыдущих шагах данным наличие оптимального значения наиболее вероятно, перспективным интервалом.

Параметрами случайного поиска (СП), которыми может варьировать пользователь, являются: n_{step} – число шагов, $\varepsilon = 1/2\varepsilon$ – точность, с которой ищется минимум, а также параметры $p_{\min}$ и $q_{\min}$.

Параметр $q_{\min}$ определяет n - мерную площадь $s_{\min}$ такую, что на всем протяжении процесса поиска, пока $s_k > s_{\min}$, плотность моделирования вне «перспективной» области h_k не меняется. Когда же s_k становится меньше $s_{\min}$, то h_k начинает стремиться к нулю, т.е. поиск с нарастающей интенсивностью ведется внутри I^k.

От величины $p_{\min}$ во многом зависит вероятность того, что минимум в процессе поиска окажется внутри I^k и значение высоты h_k. Таким образом, параметры $p_{\min}$ и $q_{\min}$ в котором смысле позволяют задать соотношение «локальных» и «глобальных» шагов к их общему количеству и тем самым определить общее поведение СП.

В процессе поиска происходит накопление информации о характере поведения функции цели, поэтому ширину «перспективного» интервала I_i^k логично с каждым шагом сужать от $p_1 = 1$ до $p_{k_m} = p_{\min} < 1,$ что влечет уменьшение вероятности попадания p_k в I^k. Так как предполагается, что процесс поиска сходится к глобальному минимуму, то опять-таки логично предположить, что

$$\lim_{k\to\infty} p_k = 1, \quad \lim_{k\to\infty} q_k = 0, \tag{3.13}$$

т.е. считается, что далее, начиная с $p_{k_m} = p_{\min}$, функция p_k увеличивается до единицы.

- **Модификация алгоритма на базе использования логистической кривой**

Рассматривается модификация алгоритма случайного поиска, основанная на использовании логистического уравнения [3]:

$$\frac{dv}{dt} = \mu\left(1 - \frac{v}{V_\infty}\right)v, \tag{3.14}$$

где v - объем «перспективной» области определения задачи оптимизации, $v = (2q)^n, q$ - ее радиус, n размерность задачи оптимизации, $V_{\lim} = \lim_{t\to\infty} v(t) = const,$ μ - мальтузианский параметр, характеризующий в нашем случае скорость изменения знаний о решаемой задаче оптимизации и таким образом влияющий на адаптацию процесса СП. Поиск осуществляется в единичном n - мерном гиперкубе.

Решаем уравнение, находим:

$$v = \frac{1}{\frac{1}{V_\infty} + \left(\frac{1}{V_0} - \frac{1}{V_\infty}\right)e^{-\mu t}}, \tag{3.15}$$

где $V_0 = v(0)$. При малых значениях t знания возрастают экспоненциально, при больших – приближаются к определенному

пределу $V_\infty = const$. В нашей задаче q изменяется от 0.5 до 0, поэтому положим $V_\infty = 1,\ 0 \le V_0 \le V_\infty$. Тогда

$$q = \frac{1}{2}\left(1 - \frac{1}{1 + \left(\frac{1}{V_0} - 1\right) e^{-\mu k_{step} / n_{step}}}\right), \tag{3.16}$$

где k_{step} - номер шага.

Выведем $V_0 = V_0(\mu, \varepsilon), \mu = \mu(V_0, \varepsilon)$ (в предположении, что точность поиска совпадает с радиусом окрестности глобального минимума). На последнем шаге $2q = \varepsilon$, получим:

$$V_0 = \frac{1}{\frac{\varepsilon}{1-\varepsilon} e^{\mu} + 1} \tag{3.17}$$

$$\mu = -\ln\left(\frac{\varepsilon}{1-\varepsilon}\frac{V_0}{1-V_0}\right). \tag{3.18}$$

Таким образом, в работе рассматривается показательный закон изменения «перспективного» интервала, при котором его размер на каждом шаге уменьшается в $k' = const$ раз $(q = qk')$ и логистический закон с параметром μ (рис. 3.11).

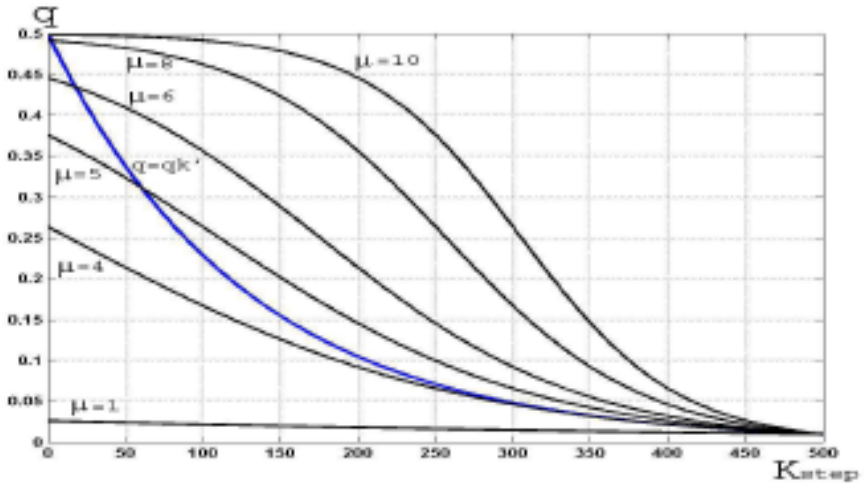


Рис. 3.8. Показательный и логистический закон изменения

Рассмотрим алгоритм поиска при крайних параметрах логистической кривой.

- Возьмем радиус «перспективной» области q почти не меняющимся, т.е. $2q = const \approx \varepsilon(\mu \to 0)$. С каждым шагом происходит равномерное бросание точки в пределах n - мерного шара радиуса - случайный выбор направления, в котором перемещаемся. Таким образом и происходит спуск к некоторому минимуму. Вероятность в этом случае оказывается выше, чем в случае равномерно распределенного СП и зависит от вида функции цели.
- Возьмем $q : 2q = const \approx \varepsilon(\mu \to 0)$. В этом случае сужения промежутка практически не будет, и СП будет эквивалентен равномерно распределенному СП (где точки, в которых вычисляется целевая функция Φ, являются независимыми реализациями случайного элемента с равномерным на X распределением).

Следовательно, предполагается, что вероятность в зависимости от параметра μ увеличивается от (1)-го случая, затем уменьшается до (2)-го, и существует параметр, при котором вероятность принимает максимальное свое значение. Назовем его оптимальным параметром μ. Далее производится подбор этого параметра.

Было замечено, что для некоторых функций при разных n_{step} существуют разные оптимальные параметры.

Для унимодальных функций с увеличением n_{step} параметр μ уменьшается с 4 до 3 (см.рис.3.9).

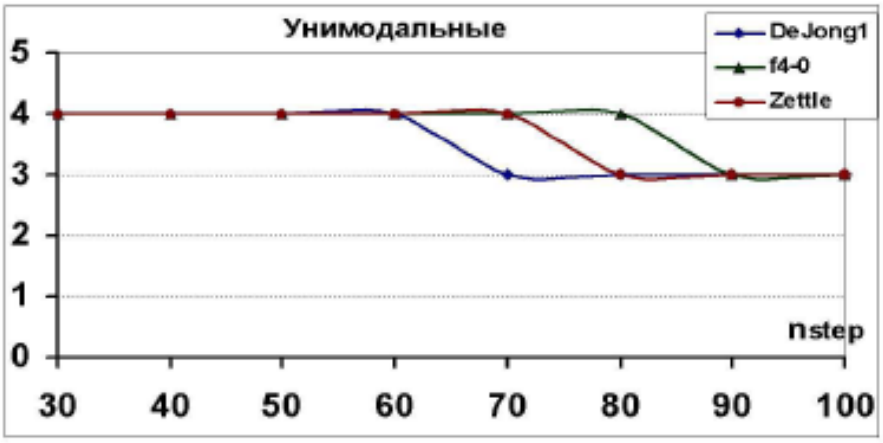


Рис.3.9. График оптимального параметра μ в зависимости от n_{step} для унимодальных функций

В случае овражных функций и некоторых многоэкстремальных функций оптимальное μ наоборот увеличивается. Данное направление требует дальнейших исследований.

- **Нахождение оптимального параметра**

В качестве критерия эффективности выбрана вероятность нахождения глобального минимума в зависимости от значений параметра μ логистической кривой.

За оценку вероятности P сходимости СП к Φ^* выбрано отношение числа случаев, в которых найденные значения переменных x^* целевой функции отличаются от оптимальных менее, чем на 0.01.

Требуется выявить множество параметров, универсальное для всех классов функций, при котором вероятность нахождения минимума максимальна.

Замечание: функции $f_i(x)$ имеют различную «природу». В связи с этим производится нормирование путем деления всех $f_i(x)$ на $\max_{\mu \in M} f_i(x)$.

Для каждого класса функций проводится следующая работа.

Происходит вычисление вероятности при разных значениях параметра логистической кривой для каждой тестовой функции. Для унимодальных функций найти оптимальный параметр проще – почти для всех функций он равен 4 (см. рис.3.10).

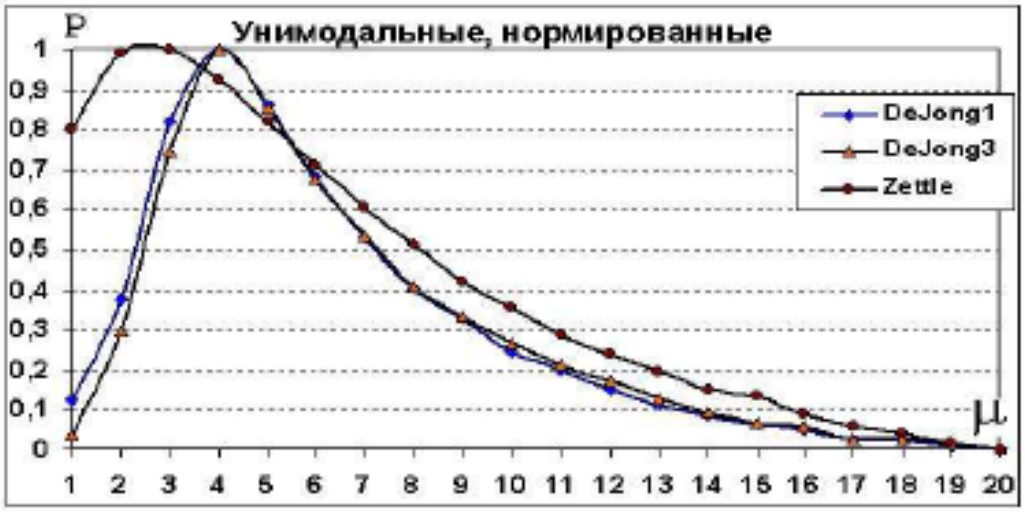


Рис.3.10. Вероятность P в зависимости от параметра μ для некоторых унимодальных функций

Для многоэкстремальных функций все гораздо сложнее. В связи с тем, что функции ведут себя по-разному, многоэкстремальность может быть самая разнообразная, то и оптимальные параметры для каждой функции свои (см. рис.3.11). Но, несмотря на это, общие тенденции прослеживаются.

Овражные функции по поведению схожи с унимодальными, следовательно, и графики для вероятностей в зависимости от μ у них будут схожими, но чуть сдвинутыми (см. рис.3.12).

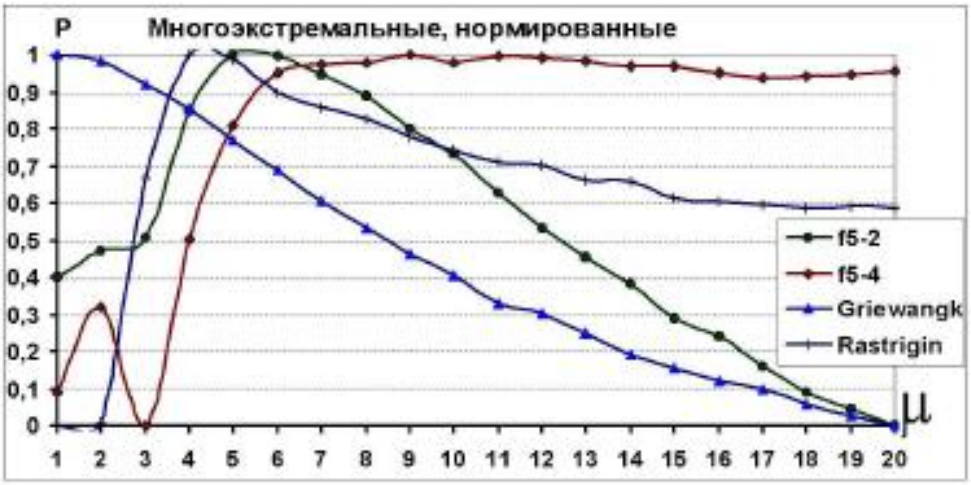


Рис.3.11. Вероятность P в зависимости от параметра μ для некоторых многоэкстремальных функций

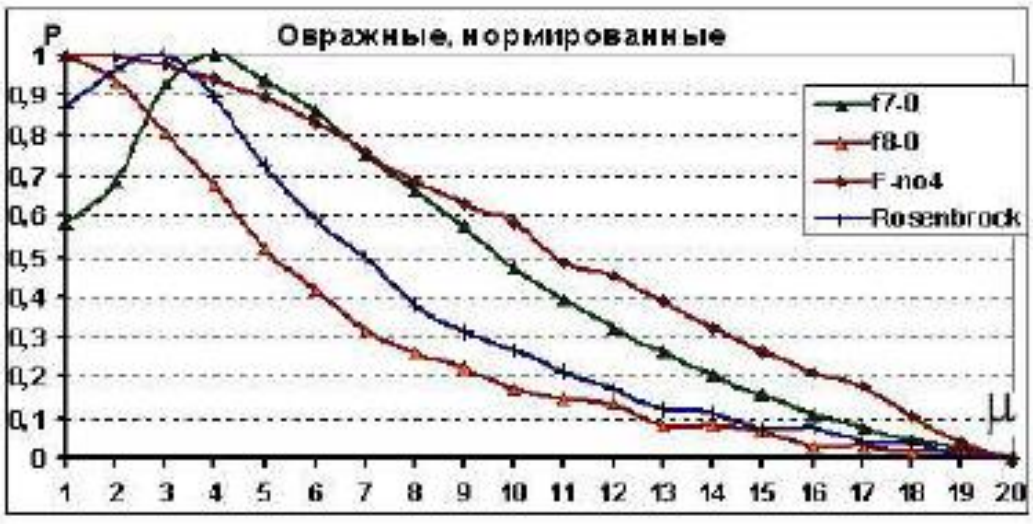


Рис.3.12. Вероятность P в зависимости от параметра μ для некоторых овражных функций.

Заметим, что для почти всех унимодальных функций наилучшим параметром является $\mu = 4$. Это значит, что выявление глобальных тенденций функции не так важно, и при таком параметре «перспективная» область вполне спускается в единственный глобальный минимум за n_{step} шагов.

Аналогично для овражных функций. Но в этом случае уже значение оптимального параметра μ уменьшается. То есть здесь уже тратятся шаги не на уточнение минимума, а на спуск к нему.

В случае многоэкстремальных функций, судя по оптимальным параметрам, видно, что μ уже немного выше, чем для унимодальных и овражных функций. Это связано с тем, что здесь выявление глобальных тенденций уже более важно, что хорошо видно на примере функции *f5-4*. Причем, чем сложнее функция, тем выше параметр μ.

Если в качестве критерия взять зависимость дисперсии от значений параметра логистической кривой, то в результате видно, что оптимальные параметры μ (т.е. такие параметры, при которых дисперсия принимает наименьшее значение) примерно такие же, как и для критерия вероятности, хотя в случае многоэкстремальных функций формально выделить оптимальные параметры сложно, но общие тенденции и тоже прослеживаются. Следовательно, для вычисления универсального параметра можно пользоваться и критерием дисперсии.

Таким образом видно, что каждый класс функций ведет себя своим особенным образом и подходят для него свои особенные параметры. Следовательно, ставится задача о подборе универсального параметра для всех классов функций.

Задача принятия решения здесь является несложной, т.к., во-первых, количество альтернатив не слишком велико, а во-вторых, для альтернатив прослеживаются некоторые общие тенденции относительно их значений.

Далее происходит упорядочение значений параметра μ внутри каждого класса функций.

На последнем этапе осуществляется поиск хороших параметров для всех классов функций с помощью методов принятия решений [34].

Таким образом получили оптимальное упорядочение параметров по значениям вероятностей для каждого класса тестовых

функций. Критериями в данном случае будут выступать классы функций: *F1*–унимодальные функции, *F2*– многоэкстремальные функции, *F3*–овражные функции. Альтернативами служат значения параметра μ.

Замечание: граница берется одинаковой для всех критериев. Выбор границы для каждого класса функций зависит от конкретной задачи и выходит за рамки данной работы.

Таблица 3.1

Оптимальные упорядочения значений параметра μ по значениям вероятностей по критериям *F1, F2, F3*

F1	4	3	5	6	7	2	8	9	10	1
F2	5	6	4	7	8	9	10	11	12	13
F3	3	2	1	4	5	6	7	8	9	10

3. Выберем границу для всех критериев равной 5, тогда в пересечении получим: {4,5} – параметры, которые дают наиболее приемлемые результаты. Наилучший параметр, в итоге, $\mu = \{4\}$.

Логистическую кривую с такими параметрами см. на рис.3.2.1.

То есть сразу происходит уменьшение «перспективной» области без траты шагов на выяснение глобальных тенденций функции цели.

- **Результаты**

В результате моделирования получили следующие значения вероятностей для каждой функции в случаях:

1. μ - оптимальное для данной функции,
2. Показательный закон $(q = qk')$,
3. μ - оптимальное по всем классам функций.

Унимодальные функции

Для этого класса функций использование логистической кривой оказалось равносильно показательному закону (см. рис. 3.13).

Многоэкстремальные функции

Здесь для всех функций наилучшим оказался логистический закон с соответствующими им наилучшими параметрами (см. рис. 3.14). Если сравнивать логистический закон в случае оптимального параметра и показательный закон, то разницы особой нет, но по конкретным цифрам можно сказать, что логистический закон (с универсальным параметром) не хуже. Поэтому в случае многоэкстремальных функций рекомендуется использовать логистическую кривую с универсальным параметром, полученным в

ходе решения данной задачи, либо, если имеются какие-то эмпирические данные, параметры из полученной таблицы (критерий F2).

Овражные функции

Для данного класса функций использование логистической кривой оказалось наиболее выгодным (см. рис. 3.15). Здесь, как и для остальных классов, логистический закон с соответствующими функциям оптимальными параметрами оказался наилучшим. Затем идет этот же закон при μ, оптимальным для всех классов, затем уже показательный закон. Поэтому в случае овражных функций рекомендуется использовать параметры из таблицы (критерий F3), либо универсальный параметр, полученный в ходе решения данной задачи [3,34].

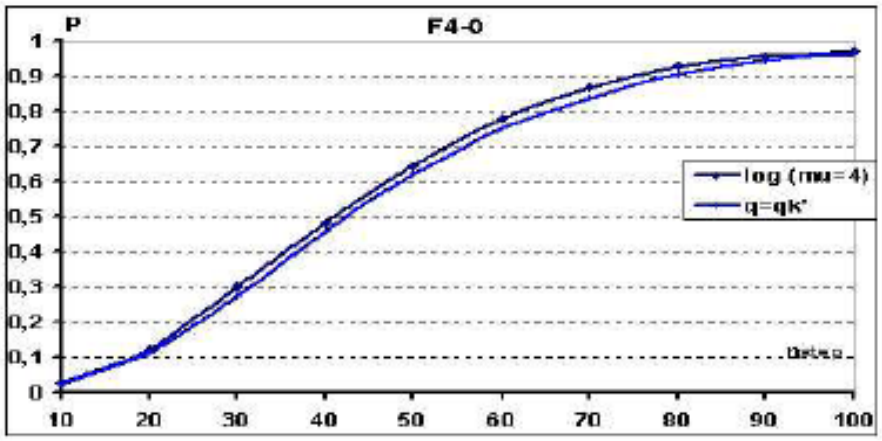


Рис.3.13. График вероятности в зависимости от n_{step} для оптимального μ для унимодальной функции *f4-0*

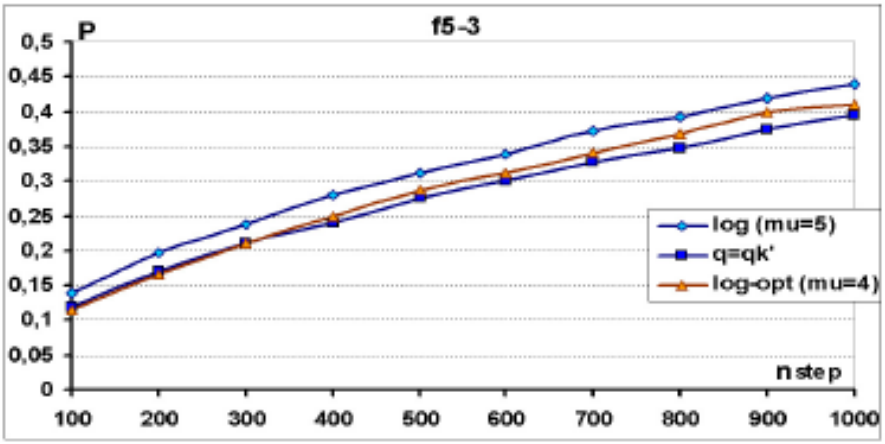


Рис.3.14. График вероятности в зависимости от n_{step} для оптимального μ для многоэкстремальной функции *f5-3*

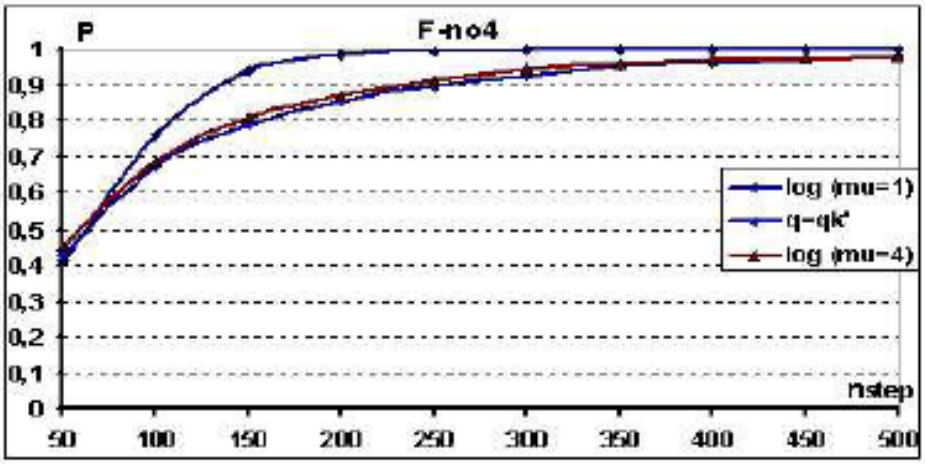


Рис.3.15. График вероятности в зависимости от n_{step} для оптимального μ для овражной функции *f-no4*

Далее можно построить график сходимости к минимуму. Строится он следующим образом: $\Phi(x)$ разбивается на сегменты (число сегментов возьмем равным 20), причем верхние и нижние границы выбираются пользователем, решающим задачу оптимизации, исходя из его эмпирических данных. Затем на каждом шаге поиска k_{step} вычисляется сегмент, в который попало значение $\Phi(x_{k_{step}})$ (см. рис. 3.16).

Из графика 3.16 получим график распределения попаданий в каждый сегмент в общем (см. рис. 3.17).

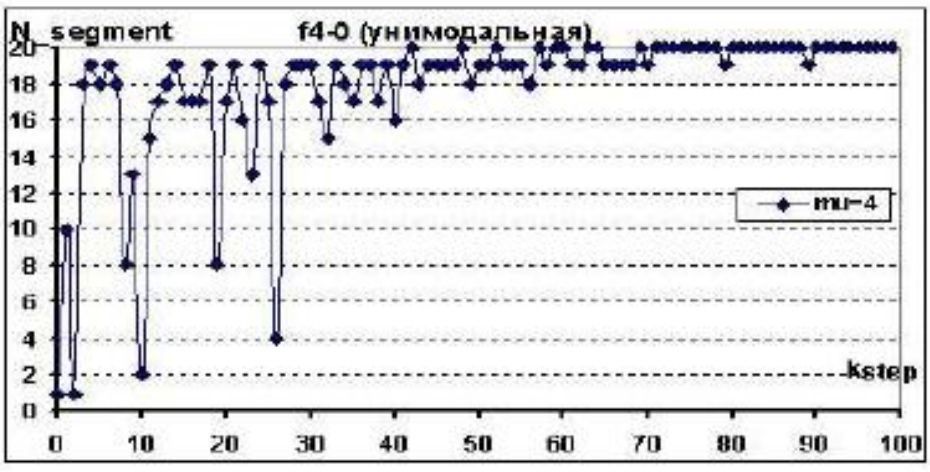


Рис.3.16. График попадания в сегменты на каждом шаге поиска в случае использования логистичиской кривой (μ =4) на примере функции *f4-0*

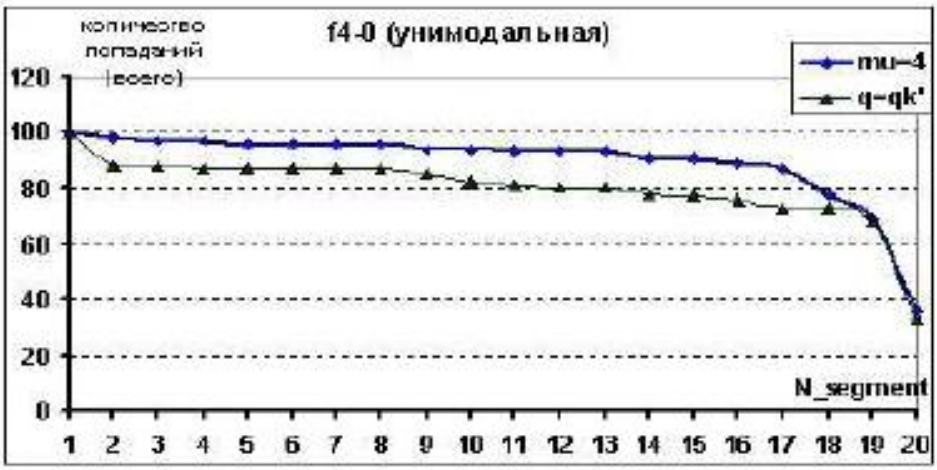


Рис.3.17. График распределения попаданий в каждый сегмент в общем

Параметры алгоритма $p_{\min}, q_{\min}$ выбираются аналогично с помощью методов принятия решений на основе исследования в работе [34].

Таким образом получилось, что для многоэкстремальных и овражных функций использование логистической кривой в алгоритме оказалось наиболее выгодным, для унимодальных функций логистический закон оказался эквивалентен показательному. В ходе решения получены универсальные для всех классов функций параметры алгоритма и разработаны рекомендации по использованию параметров для различных классов функций.

Следовательно, пользователь, решая свою задачу оптимизации, на основе личного опыта и эмпирических данных может сделать предположение о виде функции, т.е. установить ее принадлежность в той или иной степени какому-либо классу задач. Затем на базе метода принятия решений в диалоге выбрать наиболее подходящие параметры поиска.

3.3. Анализ решения задач оптимизации эволюционными методами

В общем случае оптимизация или поиск наилучшего значения (набора параметров) некоторой заданной целевой функции является достаточно сложной задачей. Сложность оптимизации обусловливается прежде всего видом целевой функции, которая может иметь глобальный, так и локальный оптимумы.

В настоящее время не существует метода оптимизации, который позволил бы решить любую задачу (был универсальным) и при этом однозначно определен как лучший среди других методов по точности решения.

По степи приближения к точному решению, а также по характеру пространства поиска задачи могут быть разделены на следующие категории.

Комбинаторные задачи – характеризуются конечным и дискретным пространством поиска. Сущность любой комбинаторной задачи можно сформулировать следующим образом: найти на множестве *X* элемент *x*, удовлетворяющий совокупности условий *K(x)*, в предположении, что пространство поиска *X* содержит некоторое конечное число различных точек.

Общие задачи без ограничений – имеют нелинейное и неограниченное пространство поиска. Методы оптимизации для таких задач обычно полагаются на правильность аналитической формулировки целевой функции. Оптимизация функции без ограничений заключается в максимизации или минимизации некоторой функции $U(x_1,...,x_p)$.

Общие задачи с ограничениями – могут быть сформулированы как задачи минимизации функции $U(x_1,...,x_p)$ при следующих ограничениях: $g_i(x_1,...,x_p) \geq 0$ для $1 \leq i \leq m, h_1(x_1,...,x_p) = 0$ для $1 \leq j \leq n$. Обычно задачи с ограничениями могут быть сведены к задачам без ограничений с помощью метода штрафов.

Если пространство поиска содержит конечное число точек, то наиболее точное решение может быть уверенно получено методом полного перебора. Этот метод имеет один очевидный недостаток – сложность вычислений, а следовательно, время, затрачиваемое на нахождение оптимального решения, существенно зависит от размерности пространства поиска. Метод перебора может быть достаточно эффективным только в небольшом пространстве поиска. А если предположить, что за одну секунду может быть выполнен миллиард операций (10^9) и при этом процесс случайного поиска начался 15 млрд. лет назад, то к настоящему времени можно было бы протестировать около 10^{27} точек из пространства поиска. Одна точка такого пространства будет представлять собой бинарную строку длиною *L* ($10^{27} \approx 2^{90}$).

Градиентные методы, являющиеся основой линейного и нелинейного, динамического программирования, а также численного анализа, более универсальны, но менее точны. При этом усложнение ландшафта пространства поиска приводит к снижению эффективности таких методов. Методы градиента не гарантируют получение единственного оптимального решения, за исключением случаю, когда пространство отображения является выпуклым и не допускает появления второстепенных вершин, плато и т.д.

С другой стороны, эвристические методы, к которым относятся генетические алгоритмы (ГА), являются наиболее универсальными, поэтому не гарантируют нахождения глобального оптимума, являющегося единственным решением задачи [171-173].

Характеристикой задача и, соответственно, основой для классификации методов оптимизации является также сложность задачи. По степени сложности однозначно выделяются следующие задачи.

Линейные задачи – сложность которых определяется как *O(n)*, где *n* – размерность входных данных задачи.

Полиномиальные задачи (*P*) – для них известен алгоритм, сложность которого составляет полином заданной, постоянной и не зависящей от размерности входной величины *n* степени.

Экспоненциальные задачи – сложность которых не менее порядка f^n, где f - константа или полином от *n*.

Однако существует большое число задач, которые не попадают ни в один из перечисленных классов. Сложность решения таких задач не может быть определена априорно. К ним относятся: оптимизация пути коммивояжера, оптимальная загрузка емкости, оптимизация маршрутов, инвестиций и т.д.

В общем случае задача оптимизации в настоящее время не может быть отнесена к какому-либо классу.

ГА являются стохастическим эвристическим методом, в котором вероятность выбора состояния $S(t+1)$ зависит от состояния *S(t)* и косвенно от предыдущих состояний. Стохастические методы позволяют решать широкий класс таких задач, поскольку не требует жестокой формализации. Следует отметить, что стохастические методы оптимизации используются для решения *NP*-сложных комбинаторных задач, т.е. таких задач, к которым сводима любая задача из класса *NP*.

Каждый из стохастических и эвристических методов имеет свои достоинства и недостатки, обусловленные формулировкой и размерностью решаемой задачи. При этом математически доказано, что для комбинаторных задач оптимизации средняя эффективность всех алгоритмов для всех возможных задач одинакова. На рис.3.18 приведена классификация эвристических и стохастических алгоритмов.

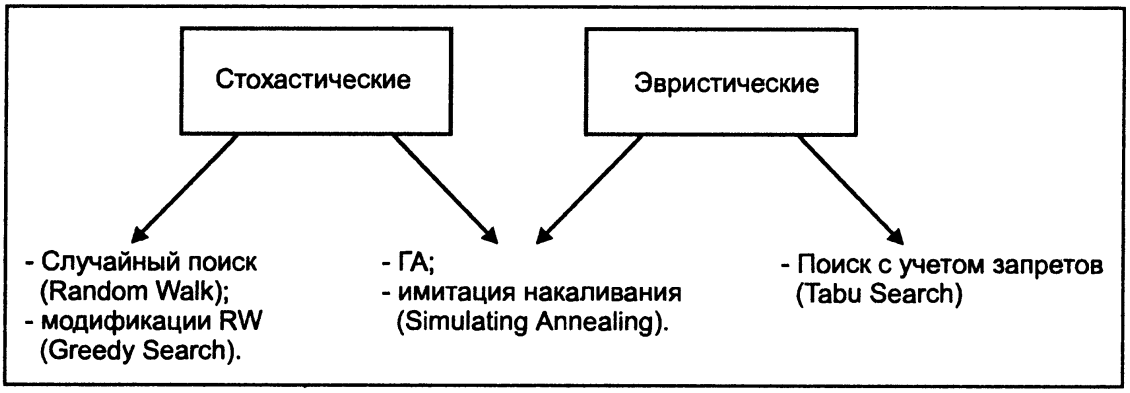


Рис.3.18. Классификация эвристических и стохастических алгоритмов

С их помощью можно найти экстремальное значение целевой функции, но не всегда можно быть уверенным, что получено значение глобального экстремума. Нахождение локального экстремума вместо глобального называется преждевременной сходимостью. Помимо проблемы преждевременной сходимости существует другая проблема — время процесса вычислений. Зачастую более точные оптимизационные методы работают очень долго.

Для решения поставленных проблем и проводится поиск новых оптимизационных алгоритмов. В настоящее время для решения многих практических задач широко применяются методы эволюционных вычислений и генетические алгоритмы, являющиеся стохастическими методами оптимизации, основанными на принципах моделирования биологической эволюции [158-177]. Развитие генетических алгоритмов началось с работ Дж. Холланда [154] и ряда других авторов, обобщивших предшествовавшие подходы к созданию алгоритмов на эволюционной основе. В настоящее время разработка и применение генетических алгоритмов является интенсивно развивающимся направлением. Благодаря

универсальности вычислительной схемы, возможностям параллельной реализации, устойчивости к шуму, генетические алгоритмы находят успешное практическое применение при решении многих сложных нелинейных многомерных задач оптимизации. В отличие от традиционных методов многопараметрической оптимизации, многие из которых часто характеризуются резким ростом вычислительных затрат при увеличении числа варьируемых параметров, генетические алгоритмы хорошо зарекомендовали себя на задачах большой размерности. В настоящее время они широко используются для решения большого круга практических задач [155], и область их применения постоянно расширяется. Генетические алгоритмы находят успешное применение при решении задач прогнозирования [156-158], управления [159-162], проектирования сложных систем [163] и др.

Генетические алгоритмы имеют следующие отличия от других методов оптимизации:

1. Генетические алгоритмы используют множество (популяцию) взаимно конкурирующих пробных решений задачи. В реальных условиях, когда целевая функция искажается случайными возмущениями, это свойство позволяет генетическим алгоритмам находить приемлемые решения и избегать остановки в локальных экстремумах.

2. При преобразовании пробных решений используются вероятностные правила, которые вносят в поиск элементы случайности. Это также позволяет решать проблему выхода из локальных экстремумов.

3. Генетические алгоритмы являются методами нулевого порядка, стратегия поиска в которых построена только на вычислении значений критерия оптимальности и не требует знания дополнительной информации о производных, константе Липшица, что характерно для градиентных и квази-ньютоновских методов. Обычно генетические алгоритмы применяются для приближенного решения задач оптимизации с большой размерностью, для которых не известно точных алгоритмов приемлемой трудоемкости. Большой интерес к генетическим алгоритмам объясняется тем что, как показывает практика, благодаря своим положительным свойствам, генетические алгоритмы часто превосходят другие стохастические и детерминированные эвристические методы и позволяют найти

близкие к оптимальным решения трудных задач поиска за существенно меньшее время [151, 152, 159].

В настоящее время весьма актуальными являются вопросы, связанные с теоретическим и экспериментальным исследованием генетических алгоритмов, изучением их свойств при решении различных классов задач.

В настоящее время при создании интеллектуальных систем управления сложными динамическими объектами начинают активно использоваться генетические алгоритмы [159]. В основном они применяются для решения оптимизационных задач, возникающих при проектировании различных классов интеллектуальных систем. Так, например, генетические алгоритмы используются для определения формы функций принадлежности нечетких регуляторов, обеспечивающих заданное качество процессов управления, для выбора топологий, архитектуры и оптимального алгоритма обучения нейронной сети, в задачах построения правил вывода в самонастраивающихся экспертных системах управления объектами и т.д. Среди других практических применений генетических алгоритмов в задачах управления можно отметить такие задачи, как: синтез оптимальных алгоритмов управления многозвенными роботами манипуляторами [161], оптимальное управление стыковкой космических аппаратов, планирование маршрутов движения транспортных средств в условиях препятствий [160], построение бесконфликтных маршрутов самолетов, идентификация сложных динамических объектов с непрерывным и дискретным временем и др.

Алгоритмы, использующие процедуры адаптации, изменяют свои параметры в процессе выполнения алгоритма, используя следующие статистические или эвристические правила:

1. Метод "адаптивной мутации" (adaptive hypermutation) [164], позволяет резко увеличить уровень мутации в том случае, когда наблюдается ухудшение качества работы алгоритма.

2. Метод "варьируемого локального поиска" {variable local search) [165], увеличивает уровень мутации после того, как зафиксировано уменьшение среднего значения функции качества.

Сначала используются небольшие мутации (изменяются только последние биты двоичной кодировки действительного числа). Если это не приводит в течение заданного времени к увеличению среднего значения функции качества в популяции, то диапазон локального поиска увеличивается путем мутации большего числа бит.

3. Методы адаптации уровня мутации и вероятности скрещивания, которые регулируются самим алгоритмом в процессе его выполнения [164, 165].

Алгоритмы, основанные на использовании дополнительных структур "памяти" (дополнительных "генов" или "хромосом") позволяют сохранять лучшие решения, полученные в прошлом, и при необходимости использовать их в будущем.

"Структурированный генетический алгоритм" основан на использовании многоуровневой структуры особи. В данном представлении гены первого уровня могут активизировать гены более низких уровней. Экспериментально было показано, что данный подход позволяет улучшить качество работы алгоритма в средах, осциллирующих между двумя различными состояниями. При этом не известно, насколько эффективно он может быть применен к системам с большим числом состояний.

В [159] предложен метод сохранения лучших особей для использования их при создании новой популяции после того, как зафиксировано изменение среды. Предполагается, что изменения происходят через фиксированное число поколений, при этом лучшие особи популяции сохраняются через заданные интервалы времени. После изменения целевой функции новая популяция частично заполняется решениями, сохраненными на предыдущих итерациях (5-10%), а остальные особи инициализируются случайным образом. В [159] было отмечено существенное улучшение качества работы алгоритма при использовании данного подхода по сравнению со случайной инициализацией. Тем не менее, при передаче в новую популяцию большего числа сохраненных решений (50-100%) или при более значительных изменениях функцйи качества работа алгоритма ухудшалась.

В [155] используется применение специальной "базы знаний" для сохранения в ней лучших особей популяции. В данном случае предполагается, что различные условия окружающей среды могут быть измерены и классифицированы. Через равные промежутки времени лучшая особь сохраняется в базе и индексируется в соответствии с текущим состоянием среды. Когда происходит изменение состояния среды, алгоритм запускается заново, при этом половину популяции составляют особи из базы знаний, которые являлись лучшими решениями при аналогичных условиях. Эксперименты показали, что использование базы знаний позволяет

эффективно использовать опыт, накопленный в прошлом, однако данный подход применим лишь в том случае, когда различные типы состояний окружающей среды могут быть классифицированы.

Очевидно, что методы третьего класса наиболее эффективны для сред, изменяющихся дискретно и периодически, и большинство из них неприменимы к задачам с непрерывно изменяющимся оптимумом. В целом, все перечисленные выше подходы являются эвристическими по существу, и их эффективность может быть разной при решении задач с различными типами изменения целевой функции.

Существуют две широко используемые методы стохастической сходимости эволюционных алгоритмов – это полное совпадения и совпадение по значению [161].

Генетические алгоритмы — это адаптивные методы поиска, которые в последнее время используются для решения задач оптимизации. В них используются как аналог механизма генетического наследования, так и аналог естественного отбора. При этом сохраняется биологическая терминология в упрощенном виде и основные понятия линейной алгебры.

Введем основные понятия, применяемые в генетических алгоритмах.

Вектор — упорядоченный набор чисел, называемых компонентами вектора. Так как вектор можно представить в виде строки его координат,то в дальнейшем понятия вектора и строки считаются идентичными.

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

Хеммингово расстояние — используется для булевых векторов и равно числу различающихся в обоих векторах компонент.

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 хеммингговой метрикой. Расстояние между двумя вершинами определяется длиной кратчайшего соединяющего их пути, измеренной вдоль ребер.

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. Каждая позиция (бит) хромосомы называется геном.

Индивидуум (генетический код, особь) — набор хромосом (вариант решения задачи). Обычно особь состоит из одной хромосомы, поэтому в дальнейшем особь и хромосома идентичные понятия.

Расстояние — хеммингово расстояние между бинарными хромосомами.

Кроссинговер (кроссовер) — операция, при которой две хромосомы обмениваются своими частями. Например, 1100&1010 → 1110&1000.

Мутация — случайное изменение одной или нескольких позиций в хромосоме. Например, 1010011 → 1010001.

Инверсия — изменение порядка следования битов в хромосоме или в ее фрагменте. Например, 1100 → 0011.

Популяция — совокупность индивидуумов.

Пригодность (приспособленность) — критерий или функция, экстремум которой следует найти.

Локус — позиция гена в хромосоме.

Аллель — совокупность подряд идущих генов.

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.

Из приведенных выше определений следует, что терминология ГА представляет собой синтез собственно генетических и искусственных понятий. Так, для понятия, заимствованного из генетики, можно предъявить его искусственный (символический) аналог. Например, хромосома и строка. В биологических системах полный генетический пакет - совокупность всех генов. 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.

В биологической терминологии говорят, что хромосома образована генами. В генетике с любым локусом связана определенная генетическая функция. Поэтому можно говорить о специализированных генах. Например, ген цвета глаз животного находится в 10 локусах, т.е. голубой цвет глаз имеет 10-аллельное значение. В терминологии ГА говорят, что строки образованы значениями функции, или детекторами. Значения функции могут быть локализованы в различных позициях строки. Связь между естественной (биологической) и искусственной терминологией :

Хромосома - Строка

Ген - Значение функции, характеристика, детектор

Аллель - Возможные значения генов

Локус - Позиция в строке

Генотип - Фактическая структура. Кодированная хромосома.

Фенотип - Множество параметров, альтернативное решение, декодированная структура

Эпистаз - Нелинейность. Взаимодействие генов.

Глава 4. РЕШЕНИЕ МНОГОКРИТЕРИАЛЬНЫХ ЗАДАЧ ОПТИМИЗАЦИИ НА ОСНОВЕ ИСПОЛЬЗОВАНИЯ ГЕНЕТИЧЕСКОГО АЛГОРИТМА

4.1. Анализ особенности генетических алгоритмов

Основные принципы работы ГА заключены в следующей схеме:

1. Генерируем начальную популяцию из *n* хромосом.

2. Вычисляем для каждой хромосомы ее пригодность.

3. Выбираем пару хромосом-родителей с помощью одного из способов отбора.

4. Проводим кроссинговер двух родителей с вероятностью p_c, производя двух потомков.

5. Проводим мутацию потомков с вероятностью p_m.

6. Повторяем шаги 3–5, пока не будет сгенерировано новое поколение популяции, содержащее *n* хромосом.

7. Повторяем шаги 2–6, пока не будет достигнут критерий окончания процесса.

Критерием окончания процесса может служить заданное количество поколений или схождение (convergence) популяции.

Схождением называется такое состояние популяции, когда все строки популяции почти одинаковы и находятся в области некоторого экстремума. В такой ситуации кроссинговер практически никак не изменяет популяции, так как создаваемые при нем потомки представляют собой копии родителей с перемененными участками хромосом. Вышедшие из этой области за счет мутации особи склонны вымирать, так как чаще имеют меньшую приспособленность, особенно если данный экстремум является глобальным максимумом. Таким образом, схождение популяции обычно означает, что найдено лучшее или близкое к нему решение.

Основными операторами ГА являются кроссинговер, мутация, выбор родителей и селекция (отбор хромосом в новую популяцию). Вид оператора играет важную роль в реализации и эффективности ГА. Существуют основные формы операторов, чистое использование или модернизация которых ведет к получению ГА, пригодного для решения конкретной задачи. Рассмотрим некоторые из них.

4.1.1. Операторы выбора родителей

Существует несколько подходов к выбору родительской пары. Наиболее распространенными операторами выбора родителей являются следующие [174-177].

Панмиксия — самый простой оператор отбора. В соответствии с ним каждому члену популяции сопоставляется случайное целое число на отрезке $[1;n]$, где n — количество особей в популяции. Будем рассматривать эти числа как номера особей, которые примут участие в скрещивании. При таком выборе какие-то из членов популяции не будут участвовать в процессе размножения, так как образуют пару сами с собой. Какие-то члены популяции примут участие в процессе воспроизводства неоднократно с различными особями популяции. Несмотря на простоту, такой подход универсален для решения различных классов задач. Однако он достаточно критичен к численности популяции, поскольку эффективность алгоритма, реализующего такой подход, снижается с ростом численности популяции.

Инбридинг представляет собой такой метод, когда первый родитель выбирается случайным образом, а вторым родителем является член популяции ближайший к первому. Здесь «ближайший» может пониматься, например, в смысле минимального расстояния Хемминга (для бинарных строк) или евклидова расстояния между двумя вещественными векторами. Расстояние Хемминга равно числу различающихся локусов (разрядов) в бинарной строке.

При аутбридинге также используют понятие схожести особей. Однако теперь брачные пары формируют из максимально далеких особей. Последние два способа по разному влияют на поведение генетического алгоритма. Так, инбридинг можно охарактеризовать свойством концентрации поиска в локальных узлах, что фактически приводит к разбиению популяции на отдельные локальные группы вокруг подозрительных на экстремум участков ландшафта. Аутбридинг же направлен на предупреждение сходимости алгоритма к уже найденным решениям, заставляя алгоритм просматривать новые, неисследованные области. Инбридинг и аутбридинг бывает генотипным (когда в качестве расстояния берется разность значений целевой функции для соответствующих особей) и фенотипным (в качестве расстояния берется расстояние Хемминга).

Селекция состоит в том, что родителями могут стать только те особи, значение приспособленности которых не меньше пороговой

величины, например, среднего значения приспособленности по популяции. Такой подход обеспечивает более быструю сходимость алгоритма. Однако из-за быстрой сходимости селективный выбор родительской пары не подходит тогда, когда ставится задача определения нескольких экстремумов, поскольку для таких задач алгоритм, как правило, быстро сходится к одному из решений. Кроме того, для некоторых многомерных задач со сложным ландшафтом целевой функции быстрая сходимость может превратиться в преждевременную сходимость к квазиоптимальному решению. Этот недостаток может быть отчасти компенсирован использованием подходящего механизма отбора, который бы «тормозил» слишком быструю сходимость алгоритма. Пороговая величина в селекции может быть вычислена разными способами. Поэтому в литературе по ГА выделяют различные вариации селекции. Наиболее известные из них — это турнирный и рулеточный (пропорциональный) отборы.

При турнирном отборе (tournament selection) из популяции, содержащей N особей, выбираются случайным образом t особей, и лучшая из них особь записывается в промежуточный массив. Эта операция повторяется N раз. Особи в полученном промежуточном массиве затем используются для скрещивания (также случайным образом). Размер группы строк, отбираемых для турнира, часто равен 2. В этом случае говорят о двоичном (парном) турнире. Вообще же t называют численностью турнира. Преимуществом данного способа является то, что он не требует дополнительных вычислений.

В методе рулетки (roulette-wheel selection) особи отбираются с помощью N «запусков» рулетки, где N - размер популяции. Колесо рулетки содержит по одному сектору для каждого члена популяции. Размер i-го сектора пропорционален вероятности попадания в новую популяцию $P(i)$.

Ожидаемое число копий i-ой хромосомы после оператора рулетки определяются по формуле $N_i = P(i)N$. При таком отборе члены популяции с более высокой приспособленностью с большей вероятностью будут чаще выбираться, чем особи с низкой приспособленностью). Другие способы отбора можно получить на основе модификации выше приведенных. Так, например, в рулеточном отборе можно изменить формулу для вероятности попадания особи в новую популяцию.

4.1.2. Рекомбинация (воспроизведение)

Оператор рекомбинации применяют сразу же после оператора отбора родителей для получения новых особей-потомков. Смысл рекомбинации заключается в том, что созданные потомки должны наследовать генную информацию от обоих родителей. Различают дискретную рекомбинацию и кроссинговер[174].

4.1.2.1. Дискретная рекомбинация

Дискретная рекомбинация (Discrete recombination) в основном применяется к хромосомам с вещественными генами. Основными способами дискретной рекомбинации являются собственно дискретная рекомбинация, промежуточная, линейная и расширенно линейная рекомбинации.

Промежуточная рекомбинация (Intermediate recombination) применима только к вещественным переменным, но не к бинарным. В данном методе предварительно определяется числовой интервал значений генов потомков, который должен содержать значения генов родителей. Потомки создаются по следующему правилу:

Потомок = Родитель 1 + α·(Родитель 2 — Родитель 1), где множитель α — случайное число на отрезке $[-d,1+d]$. Как отмечают сторонники этого метода, наиболее оптимальное воспроизведение получается при $d = 0{,}25$.

При промежуточной рекомбинации возникают значения генов, отличные от значения генов особей-родителей. Это приводит к возникновению новых особей, пригодность которых может быть лучше, чем пригодность родителей. В литературе такой оператор рекомбинации иногда называется дифференциальным скрещиванием.

Линейная рекомбинация (Line recombination) отличается от промежуточной тем, что множитель α выбирается для каждого потомка один раз.

4.1.2.2. Кроссинговер (бинарная рекомбинация)

Рекомбинацию бинарных строк принято называть кроссинговером (кроссовером) или скрещиванием [174-177].

Одноточечный кроссинговер (Single-point crossover) моделируется следующим образом. 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 хромосомы (точка разрыва), в которой обе хромосомы делятся на две части и обмениваются ими.

Такой тип кроссинговера называется одноточечным, так как при нем родительские хромосомы разделяются только в одной случайной точке.

Также применяется двух- и N-точечный кроссинговер.

В двухточечном кроссинговере (и многоточечном кроссинговере вообще) 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. Следовательно, двухточечный кроссинговер решает ту же самую задачу, что и одноточечный, но более полно. Хромосома, рассматриваемая как цикл, может содержать большее количество стандартных блоков, так как они могут совершить «циклический возврат» в конце строки. В настоящий момент многие исследователи соглашаются, что двухточечный кроссинговер вообще лучше, чем одноточечный.

Для многоточечного кроссинговера (Multi–point crossover), выбираем m точек разреза. Точки разреза выбираются случайно 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.

Применение многоточечного кроссинговера требует введения нескольких переменных (точек разреза), и для воспроизведения выбираются особи с наибольшей приспособленностью.

Одноточечный и многоточечный кроссинговер определяют точки разреза, которые разделяют особи на части. Однородный кроссинговер (Uniform crossover) создает маску (схему) особи, в каждом локусе которой находится потенциальная точка кроссинговера. Маска кроссинговера имеет ту же длину, что и скрещивающиеся особи.

Таким образом, гены маски представляют собой случайные двоичные числа (0 или 1). Согласно этим значениям, геном потомка становится первая (если ген маски = 0) или вторая (если ген маски = 1) особь-родитель.

Однородный кроссинговер очень похож на многоточечный, но строка 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.

Триадный кроссинговер (Triadic crossover). Данная разновидность кроссинговера отличается от однородного тем, что после отбора пары родителей из остальных членов популяции случайным образом выбирается особь, которая в дальнейшем используется в качестве маски. Далее 10 % генов маски мутируют. Затем гены первого родителя сравниваются с генами маски: если гены одинаковы, то они передаются первому потомку, в противном случае на соответствующие позиции хромосомы потомка переходят гены второго родителя. Генотип второго потомка отличается от генотипа первого тем, что на тех позициях, где у первого потомка стоят гены первого родителя, у второго потомка стоят гены второго родителя и наоборот.

Перетасовочный кроссинговер (Shuffler crossover). В данном алгоритме особи, отобранные для кроссинговера, случайным образом обмениваются генами. Затем выбирают точку для одноточечного кроссинговера и проводят обмен частями хромосом. После скрещивания созданные потомки вновь тасуются. Таким образом, при каждом кроссинговере создаются не только новые потомки, но и модифицируются родители (старые родители удаляются), что позволяет сократить число операций по сравнению с однородным кроссинговером.

Кроссинговер с уменьшением замены (Crossover with reduced surrogate). Оператор уменьшения замены ограничивает кроссинговер, чтобы всегда, когда это возможно, создавать новые особи. Это осуществляется за счет ограничения на выбор точки разреза: точки разреза должны появляться только там, где гены различаются.

Как было показано выше, кроссинговер генерирует новое решение (в виде особи-потомка) на основе двух имеющихся, комбинируя их части. Поэтому число различных решений, которые могут быть получены кроссинговером при использовании одной и той же пары готовых решений, ограничено. Соответственно, пространство, которое ГА может покрыть, используя лишь кроссинговер, жестко зависит от генофонда популяции. Чем

разнообразнее генотипы решений популяции, тем больше пространство покрытия. При обнаружении локального оптимума соответствующий ему генотип будет стремиться занять все позиции в популяции, и алгоритм может сойтись к ложному оптимуму. Поэтому в генетическом алгоритме важную роль играют мутации. Существует несколько способов мутирования генов. Вопрос о выборе подходящего оператора мутации решается в рамках поставленной задачи [87,119].

4.1.3. Мутация

После процесса воспроизводства происходят мутации (mutation). Данный оператор необходим для «выбивания» популяции из локального экстремума и препятствует преждевременной сходимости. Это достигается за счет того, что изменяется случайно выбранный ген в хромосоме.

Так же как и кроссинговер, мутации могут проводиться не только по одной случайной точке. Можно выбирать для изменения несколько точек в хромосоме, причем их число также может быть случайным. Используют и мутации с изменением сразу некоторой группы подряд идущих точек.

4.1.4. Операторы отбора особей в новую популяцию

Для создания новой популяции можно использовать различные методы отбора особей [119,174].

Отбор усечением (Truncation selection). 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. Из-за того, что в этой стратегии используется отсортированная популяция, время ее работы может быть большим для популяции большого размера и зависеть также от алгоритма сортировки.

Элитарный отбор (Elite selection). Создается промежуточная популяция, которая включает в себя как родителей, так и их потомков. 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. Использование стратегии элитизма оказывается весьма полезным для эффективности ГА, так как не допускает потерю лучших решений. К примеру, если популяция сошлась в локальном максимуме, а мутация вывела одну из строк в область глобального, то при предыдущей стратегии весьма вероятно, что эта особь в результате скрещивания будет потеряна, и решение задачи не будет получено. Если же используется элитизм, то полученное хорошее решение будет оставаться в популяции до тех пор, пока не будет найдено еще лучшее.

Отбор вытеснением (Exclusion selection). В данном отборе 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.

Метод Больцмана, или метод отжига (Bolzman selection). В данном методе вероятность отбора в новую популяцию зависит от управляющего параметра — температуры *T*.

4.1.5. Параметры ГА

Вероятность кроссинговера обычно выбирается достаточной высокой 80–95 %. Однако, в некоторых задачах наилучший результат достигается при кроссинговере с вероятностью 60 %.

Вероятность мутации должна быть мала: 0,5–1 %.

Как это ни странно, очень большой размер популяции обычно не приводит к хорошим результатам (скорость сходимости алгоритма не увеличивается). Оптимальный размер популяции — 20–30 особей, однако в некоторых задачах, как пишут исследователи, требуется 50–100 особей. Исследования показывают, что оптимальный размер популяции зависит от размера кодовых строк (хромосом). Так, для алгоритма с 32-битовыми хромосомами размер популяции будет больше, чем для алгоритма с 16-битовыми хромсомами.

В основном используют рулеточный отбор, но и иногда лучше применить отбор с усечением. Существует много других методов, меняющих параметры отбора в течение всей работы ГА. Элитизм применяется, если не используются другие методы, сохраняющие найденное хорошее решение. Выбор способа кодирования обусловлен поставленной задачей и размером объекта поиска [87,119,173].

Операторы жизненного цикла (кроссинговера и мутации) определяются выбранным кодированием.

4.1.6. Модернизация ГА

Одной из серьезных проблем, возникающих при использовании генетических алгоритмов, является преждевременная сходимость. Не рекомендуется использовать классические ГА на маленьких популяциях, поскольку в популяциях с малым размером гены распространяются слишком быстро: все особи становятся похожими (популяция вырождается) еще до того, как найдено решение задачи. То есть новый генотип с лучшей оценкой быстро вытесняет менее хорошие комбинации генов, исключая тем самым возможность получения лучшего решения на их базе. Можно предложить три основных пути устранения преждевременной сходимости: увеличение размера популяции, применение самоадаптирующихся

генетических операторов и создание «банка» заменяемых особей [119,161].

В первом случае, увеличивая размер популяции, можно надеяться на достижение многообразия генотипа в популяции. Но, с другой стороны увеличение числа особей ведет к увеличению занимаемой памяти и времени работы алгоритма. Данный подход может быть эффективен или при параллельных вычислениях, или при наличии достаточно простой целевой функции.

Второй, и самый распространенный способ, — использование самоадаптирующихся алгоритмов — является более эффективным. Самоадаптация заключается в применении динамических мутаций. Динамические мутации в зависимости от скрещивающихся особей меняют значение вероятности мутации, тем самым становится возможным самоуправление алгоритма. В таких случаях выбирается малый размер популяции.

В третьем подходе создается массив для сохранения особей, генотип которых был утерян при формировании новых поколений, и временами эти особи добавлятся в популяцию.

Модификация стандартного варианта генетического алгоритма (Steady State GA) [19] затронула способ формирования промежуточной популяции (Mating Pool), являющейся результатом отбора (селекции) для формирования наследников с помощью генетических операторов. SSGA не формируют промежуточную популяцию как стандартный генетический алгоритм, а осуществляют последовательно выбор пары наилучших индивидов, применяя к ним генетические операторы с целью формирования наследников, которые заменяют худшие индивиды популяции. Данная модификация генетического алгоритма – Steady State GA состоит из следующих этапов [82,87,159]:

Шаг 0. Определение генетического представления задачи

Шаг 1. Создание первоначальной популяции $P(0) = x_1^0,...,x_N^0,\ t = 0$.

Шаг 2. Вычисление относительной (нормализованной) степени пригодности $f_H(x_i) = f(x_i) / \sum_{i=1}^{N} f(x_i) / N$.

Шаг 3. Выбор пары из лучших индивидов. Выбор худшего индивида. Применение скрещивания и мутация к выбранной паре лучших индивидов. Результат замещает худший индивид.

Шаг 4. $t = t + 1$, возврат к шагу 2.

При проектирования генетического алгоритма могут быть выгодно использованы знания, полученные селекционерами в области искусственной селекции. Генетические алгоритмы селекционеров (ГАС) моделируют именно искусственную селекцию ГАС, где под виртуальным селекционером понимается некоторый механизм селекции, который и является основным отличием ГАС от стандартного генетического алгоритма.

Генетический алгоритм селекционеров состоит из следующих этапов:

Шаг 0. Определение генетического представления задачи.

Шаг 1. Создание первоначальной популяции индивидов $P(0)$ размером $N, t = 0$.

Шаг 2. Виртуальный селекционер отбирает $T\%$ популяции для создания потомков. Это дает набор отобранных родителей.

Шаг 3. Формирование случайном образом из данного набора $N/2$ пар. Применение к каждой паре скрещивания и мутация, формируя новой популяции $P(t+1)$.

Шаг 4. $t = t+1$, возврат к шагу 2.

Шаг 5. Возврат к шагу 3.

Селекция основывается преимущественно на статистических методах, которые позволяют произвести теоретический анализ и прогнозировать эффективность механизмов селекции, мутации и рекомбинации с помощью введенных уравнений селекции, реакции на селекцию и понятия наследственности.

Еще одна модификация генетического алгоритма затрагивает решение многокритериальных задач. Многокритериальный генетический алгоритм (МГА) также является модификацией стандартного генетического алгоритма и отличается способом селекции, поскольку при отборе пар родителей в этом случае используется не один, а несколько критериев. При этом предлагается большое число вариантов схем селекции и соответственно вариантов МГА. Сравнительные оценки показывают, что по эффективности VEGA имеет средние показатели, однако не оценивалась вычислительная сложность для различных вариантов МГА, по которой VEGA может существенно улучшить свои показатели[119]].

Многокритериальный генетический алгоритм состоит из следующих этапов:

Шаг 0. Определение генетического представления задачи

Шаг 1. Создание первоначальной популяции $P(0) = x_1^0,...,x_N^0$, $t = 0$.

Шаг 2. Последовательное выполнение шагов 2.1-2.3.

Шаг 2.1. Вычисление значения степени пригодности каждого индивида по критерию $i = 1,...,k$.

Шаг 2.2. Для j от 1 до N/k осуществление селекции индивида из популяции в промежуточную популяцию.

Шаг 2.3. Возврат к шагу 2.1, если $l < k$.

Шаг 3. Формирование случайном образом из данного набора $N/2$ пар. Применение к каждой паре скрещивания, а также других генетических операторов, таких как мутация для формирования новой популяции $P(t+1)$.

Шаг 4. $t = t+1$, возврат к шагу 2.

Стандартный генетический алгоритм представляет собой строго синхронизованный последовательный алгоритм, который в условиях большого пространства поиска или сложного пространства поиска может быть неэффективен по критерию времени. Эту проблему позволяет решить другой вид генетического алгоритма – параллельный генетический алгоритм (ПГА). Следует отметить, что любая последовательная модификация стандартного генетического алгоритма может быть преобразована в параллельную.

По степени распараллеливания можно выделить следующие типы параллельных генетических алгоритмов:

1) ПГА на базе популяции;
2) ПГА на базе подпопуляций;
3) ПГА на базе индивидов.

ПГА на базе популяции сохраняет стандартную структуру генетического алгоритма, работающего с целой популяцией, распараллеливание реализуется на этапе скрещивания и мутации. По степени распараллеливания процессов можно выделить следующие модели [174-177]:

1. Синхронная модель «ведущий-ведомый», где главный процесс хранит целую популяцию в собственной памяти, выполняет селекцию, скрещивание и мутацию, но оставляет вычисление степени пригодности новых индивидов подчиненным процессам;

2. Полусинхронная модель «ведущий-ведомый», где новый индивид обрабатывается по мере освобождения одного из процессов;

3. Асинхронная параллельная модель, где индивиды популяции хранятся в общей памяти, к которой можно обращаться

параллельным процессам. Каждый процесс выполняет оценку степени пригодности, а также генетические операции.

Каждый процесс работает независимо от других. Единственное отличие между этой моделью и стандартным генетическим алгоритмом заключается в механизме селекции [177]. Очевидным в этом случае является вариант использования $N/2$ параллельных процессоров при популяции в N индивидов. Тогда каждый процессор дважды случайным образом выбирает два индивида из общей памяти и оставляет лучшего. Два выбранных индивида затем подвергаются скрещиванию, мутации и оценке степени пригодности. Возникающие в результате наследники размещаются в общей памяти.

Распределенный параллельный генетический алгоритм состоит из следующих этапов:

Шаг 0. Определение генетического представления задачи

Шаг 1. Создание первоначальной популяции индивидов и разделение на подпопуляции $SP_1,...,SP_N$.

Шаг 2. Формирование структуры подпопуляций.

Шаг 3. Для $SP_1,...,SP_N$ - выполнение параллельно шагов 3.1-3.3.

Шаг 3.1. Применение в течение m поколений селекции и генетических операторов.

Шаг 3.2.Перемещение k хромосом в соседние подпопуляции.

Шаг 3.3. Получение хромосом из соседних подпопуляций.

Шаг 4. Возврат к шагу 3.

Особенность ПГА на базе подпопуляций заключается в использовании независимых конкурирующих подпопуляций, которые обмениваются индивидами с заданной частотой. При этом каждый процессорный блок выполняет последовательный генетический алгоритм с собственной подпопуляцией, при условии максимизации одной общей для всех функции степени пригодности. В этом случае для обмена индивидами должна быть определена структура связей подпопуляций. С точки зрения оценки и сравнения эффективности может быть рассмотрен вариант распределенной модели, в которой обмен индивидами не осуществляется. Результаты, представленные в [162], свидетельствуют о большей эффективности распределенного ПГА по сравнению с этим частным случаем, а также со стандартным генетическим алгоритмом.

Существенным недостатком модели может стать снижение степени разнообразия при интенсивном обмене индивидами. С другой стороны, недостаточно частое перемещение может привести к

преждевременной сходимости подпопуляций. При построении такой модели важно определить следующее:

- Связи между процессорами для обмена индивидами;
- Частоту обмена индивидами (оптимальной является частота обмена через 20 поколений [162]);
- Степень перемещения или число обмениваемых индивидов (оптимальным является 20% подпопуляции [87.119]);
- Способ селекции индивида для обмена;
- Критерий, по которому полученный индивид сможет заменить члена подпопуляции.

С точки зрения времени и даже числа поколений, затрачиваемых на решение задачи, ПГА эффективнее стандартного генетического алгоритма, но при этом некоторые задачи могут быть слишком простыми для ПГА. Параллельный поиск имеет смысл в том случае, если пространство поиска большое и сложное [164]. Увеличение числа процессоров в данной модели улучшает скорость сходимости, но не качество решения.

ПГА на базе индивидов имеют одну строку индивида, постоянно находящуюся в каждом процессорном элементе (ячейке). Индивиды выбирают пары и рекомбинируют с другими индивидами в их непосредственном ближайшем окружении (по вертикали горизонтали). Выбранный индивид затем совмещается с индивидом, постоянно находящимся в ячейке. В результате формируется один наследник, который может или не может заменить индивида в ячейке в зависимости от выбранной схемы замещения. Таким образом, модель является полностью распределенной и не нуждается в централизованном управлении.

Параллельный генетический алгоритм на базе индивидов состоит из следующих этапов:

Шаг 0. Определение генетического представления задачи

Шаг 1. Создание первоначальной популяции индивидов и формирование структуры популяций.

Шаг 2. Локальное повышение каждым индивидом своей производительности (hill-climbing).

Шаг 3. Выполнение каждым индивидом селекции с целью поиска пары.

Шаг 4. Применение к паре скрещивания а также других генетических операторов, таких как мутация.

Шаг 5. Локальное повышение наследником своей производительности (hill-climbing). Замещение наследником родителя в соответствии с заданным критерием качества.

Шаг 6. Возврат к шагу 3.

При работе с моделью на базе индивидов необходимо задать:

- Структуру связей ячеек;
- Схему селекции;
- Схему замещения.

Исследования этой модели показали, что для сложных задач она способна обеспечить лучшие решения, чем стандартный генетический алгоритм.

В ходе исследований в области генетических алгоритмов и эволюционных алгоритмов в целом появилось большое количество направлений, и их число непрерывно растет [158-177].

Исследования в области генетического алгоритма и эволюционных алгоритмов в целом позволяют выделить важную особенность данных методов – эффективность эволюционных алгоритмов существенно зависит от таких взаимосвязанных параметров, как вероятность применения генетических операторов, их тип и размер популяции.

4.2. Исследование сходимости генетических алгоритмов к глобальному оптимуму

Для того чтобы лучше понять функционирование генетического алгоритма, будем использовать понятие схема и сформулируем основную теорему, относящуюся к генетическим алгоритмам и называемую теоремой о схемах [173-177]. Понятие схема было введено для определения множества хромосом, обладающих некоторыми общими свойствами, т.е. подобных друг другу. Если биты принимают значения 0 или 1 (рассматриваются хромосомы с двоичным алфавитом), то схема представляет собой множество хромосом, содержащих нули и единицы на некоторых заранее определенных позициях. При рассмотрении схем удобно использовать расширенный алфавит {0, 1, *}, в который помимо 0 и 1 введен дополнительный символ *, обозначающий любое допустимое значение, т.е. 0 или 1; символ * в конкретной позиции означает «все равно».

Считается, что хромосома принадлежит к данной схеме, если для каждой j-й позиции, $j = 1,2,...,L$, где L - длина хромосомы;

символ, занимающий j-ю позицию схемы, причем символу * соответствуют как 0, так и 1.

Генетический алгоритм базируется на принципе трансформации наиболее приспособленных хромосом. Пусть $P(0)$ означает исходную популяцию особей, а $P(k)$ - текущую популяцию (на k-й итерации алгоритма). Из каждой популяции $P(k),\ k = 0,1,...$ методом селекции выбираются хромосомы с наибольшей приспособленностью $M(k)$. Далее, результате объединения хромосом из популяции $M(k)$ в родительские пары и выполнения операции скрещивания с вероятностью p_c, а также операции мутации с вероятностью p_m формируется новая популяция $P(k+1)$, в которую входят потомки хромосом из популяции $M(k)$.

Следовательно, для любой схемы, представляющей хорошее решение, было бы желательным, чтобы количество хромосом, соответствующих этой схеме, возрастала с увеличением количества итерации k.

На соответствующее преобразование схем в генетическом алгоритме оказывают влияние 3 фактора: селекция хромосом, скрещивание и мутация. Проанализуем воздействие каждого из них с точки зрения увеличения ожидаемого количества представителей отдельно взятой схемы.

Обозначим рассматриваемую схему символом S, а количества хромосом популяции $P(k)$, соответствующих этой схеме – $c(S,k)$. Следовательно, $c(S,k)$ можно считать количеством элементов (т.е. мощностью) множества $P(k) \cap S$.

Начнем с исследования влияния селекции. При выполнении этой операции хромосомы из популяции $P(k)$ копируются в родительскую хромосому $M(k)$ с вероятностью, определяемой выражением $P_s(ch_j) = F(ch_j) \Big/ \sum_{i=1}^{N} F(ch_j)$, где N - численность популяции. Пусть $F(S,k)$ обозначает среднее значение функции приспособленности хромосом из популяции $P(k)$, которые соответствуют схеме S. Если

$$P(k)S = \{ch_1,...,ch_{c(S,k)}\},$$

то

$$F(S,k)=\frac{\sum_{j=1}^{c(S,k)}F(ch_j)}{c(S,k)}. \qquad (4.1)$$

Величина $F(S,k)$ также называется приспособленностью схемы S на k-й итерации.

Пусть $\Im(k)$ обозначает сумму значений функций приспособленности хромосом из популяции $P(k)$ мощностью N, т.е.

$$\Im(k)=\sum_{i=1}^{N}F(ch_i^{(k)}).$$

Обозначим через $\overline{F}(k)$ среднее значение функции приспособленности хромосом этой популяции, т.е.

$$\overline{F}(k)=\frac{1}{N}\Im(k).$$

Пусть $ch_j^{(k)}$ обозначает элемент из родительской хромосомы $M(k)$. Для каждого $ch_j^{(k)}\in M(k)$ и для каждого $j=1,...,c(S,k)$ вероятность того, что $ch_j^{(k)}=ch_j$ определяется отношением $F(ch_j)/F(k)$. Поэтому ожидаемое количество хромосом в популяции $M(k)$, которые равны ch_j составит

$$N\frac{F(ch_j)}{\Im(k)}=\frac{F(ch_j)}{\overline{F}(k)}.$$

Таким образом, ожидаемое количество хромосом из множества $P(k)\cap S$, отобранных для включения в родительскую хромосому $M(k)$, будет равно

$$\sum_{i=1}^{c(S,k)}\frac{F(ch_i)}{\overline{F}(k)}=c(S,k)\frac{F(S,k)}{\overline{F}(k)},$$

что следует из выражения (4.1).

Поскольку каждая хромосома из родительской популяции $M(k)$ одновременно принадлежит популяции $P(k)$, то хромосомы, составляющие множества $M(k)\cap S$ - это те же самые хромосомы, которые были отобраны из множества $P(k)\cap S$ для включения в популяцию $M(k)$. Если количество хромосом родительской популяции $M(k)$, соответствующих схеме S (т.е. количество элементов множества $M(k)\cap S$), обозначить $b(S,k)$, то из приведенных рассуждений можно сделать следующий вывод:

Ожидаемое значение $b(S,k)$, т.е. ожидаемое значение количества хромосом из родительской популяции $M(k)$, соответствующих схеме S, определяется выражением

$$E[b(S,k)] = c(S,k)\frac{F(S,k)}{\bar{F}(k)} \ . \qquad (4.2)$$

Из этого следует, что если схема S содержит хромосомы со значением функции приспособленности, превышающим среднее значение (т.е. приспособленности хромосом из популяции $P(k)$, и поэтому $F(S,k)/\bar{F}(k) > 1$), то ожидаемое количество хромосом из родительской популяции $M(k)$, соответствующих схеме S, будет больше количества хромосом из популяции $P(k)$, соответствующих схеме S. Поэтому можно утверждать, что селекция вызывает распространение схем с приспособленностью «лучше средней» и исчезновение схем с «худшей» приспособленностью.

Прежде чем приступить к анализу влияния генетических операторов скрещивания и мутации на хромосомы из родительской популяции, определим необходимые для дальнейших рассуждений понятия порядка и охвата схемы. Пусть L обозначает длину хромосом, соответствующих схеме S.

Порядок схемы S, иначе называемый с четностью схемы и обозначаемый $o(S)$ - это количество постоянных позиций в схеме, т.е. нулей и единиц в случае алфавита $\{0,1,*\}$.

Порядок схемы $o(S)$ равен длине L за вычетом количества символов *. Порядок схемы, состоящей исключительно из символов *, равен нулю, а порядок схемы без единого символа * равен L.

Охват схемы S, называемый также длиной схемы и обозначаемый $d(S)$ - это расстояние между первым и последним постоянным символом (т.е. разность между правой и левой крайними позициями, содержащими постоянные символы).

Охват схемы $d(S)$ - это целое число из интервала $[0, L-1]$. Отметим, что охват схемы с постоянными символами на первой и последней позиции равен $L-1$. Охват схемы с единственной постоянной позицией равен нулю. Охват характеризует содержательность информации, заключенной в схеме.

В ходе анализа влияния операции скрещивания на родительский хромосомы $M(k)$ рассмотрим некоторую хромосому из множества $M(k) \cap S$, т.е. хромосому из родительской популяции, соответствующую схеме S. Вероятность того, что эта хромосома

будет отобрана для скрещивания, равна p_c. Если ни один из потомков этой хромосомы не будет принадлежать к схеме S, то это означает, что точка скрещивания должна находиться между первым и последним постоянным символом данной схемы. Вероятность этого равна $d(S)/(L-1)$. Из этого можно сделать следующие выводы о влиянии скрещивания:

Для некоторой хромосомы из $M(k) \cap S$ вероятность того, что она будет отобрана для скрещивания и ни один из ее потомков не будет принадлежать к схеме S. Ограничена сверху величиной $p_c \dfrac{d(S)}{L-1}$. Эта величина называется вероятностью уничтожения схемы S.

Для некоторой хромосомы из $M(k) \cap S$ вероятность того, что она не будет отобрана для скрещивания либо, что хотя бы один из ее потомков после скрещивания будет принадлежать к схеме S, ограничена снизу величиной $X_l = (x_{l1}, x_{l2}, ..., x_{ln})$. Эта величина называется вероятностью выживания схемы S.

Легко показать, что если данная хромосома принадлежит к схеме S и отбирается для скрещивания, а вторая родительская хромосома также принадлежит к схеме S, то оба их потомка тоже будут принадлежать к схеме S. Выводы подтверждают значимость показателя охвата схемы $d(S)$ для оценки вероятности уничтожения или выживания схемы.

Рассмотрим теперь влияние мутации на родительскую хромосому $M(k)$. Оператор мутации с вероятностью p_m случайным образом изменяет значение в конкретной позиции с 0 на 1 и обратно. Очевидно, что схема переживет мутацию только в том случае, когда все ее постоянные позиции останутся после выполнения этой операции неизменными.

Популяция родительской хромосомы, принадлежащая к схеме S (т.е. хромосома из множества $M(k) \cap S$) останется в этой схеме тогда и только тогда, когда ни один символ этой хромосомы, соответствующий постоянным символам схемы S, не изменится в процессе мутации. Вероятность такого события равна $(1-p_m)^{o(S)}$. Этот результат можно представить в форме следующего вывода о влиянии мутации:

Вероятность того, что некоторая хромосома из $M(k) \cap S$ будет принадлежать к схеме S после операции мутации, определяется выражением

$$X_i = (x_{i1}, x_{i2}, ..., x_{in}) \in \left[\underline{x_1}, \overline{x_1}\right] \times \left[\underline{x_2}, \overline{x_2}\right] \times ... \times \left[\underline{x_n}, \overline{x_n}\right].$$

Эта величина называется вероятностью выживания схемы S после мутации.

Если вероятность мутации p_m мала ($p_m << 1$), то можно считать, что вероятность выживания схемы S после мутации приближенно равна $1 - p_m o(S)$.

Эффект совместного воздействия селекции, скрещивания и мутации с учетом факта, что если хромосома из множества $M(k) \cap S$ дает потомка, соответствующего схеме S, то он будет принадлежать к $P(k+1) \cap S$, ведет к построению следующей схемы репродукции [73]:

$$E[c(S,k+1)] \geq c(S,k) \frac{F(S,k)}{\overline{F}(k)} \left(1 - p_c \frac{d(S)}{L-1}\right) (1 - p_m)^{o(S)} \qquad (4.3)$$

Зависимость (4.3) показывает, как изменяется от популяции к популяции количество хромосом, соответствующих данной схеме. Это изменение вызывается тремя факторами, представленными в правой части выражения (4.3), в частности: $F(S,k)/\overline{F}(k)$ отражает роль среднего значения функции приспособленности, $1 - p_c d(S)/(L-1)$ показывает влияние скрещивания и $(1-p_m)^{o(S)}$ - влияние мутации. Чем больше значение каждого из этих факторов, тем большим оказывается ожидаемое количество соответствий схеме S в следующей популяции. Это позволяет представить зависимость (4.3) в виде

$$E[c(S,k+1)] \geq c(S,k) \frac{F(S,k)}{\overline{F}(k)} \left(1 - p_c \frac{d(S)}{L-1} - p_m o(S)\right). \qquad (4.4)$$

Для больших популяций зависимость (4.3) можно аппроксимировать выражением

$$c(S,k+1) \geq c(S,k) \frac{F(S,k)}{\overline{F}(k)} \left(1 - p_c \frac{d(S)}{L-1} - p_m o(S)\right). \qquad (4.5)$$

Из формул (4.3) и (4.4) следует, что ожидаемое количество хромосом, соответствующих схеме S в следующем поколении, можно считать функций от фактического количества хромосом, принадлежащих этой схеме, относительной приспособленности схемы, а также порядка и охвата схемы. Заметно, что схемы с приспособленностью выше средней и с малым порядком и охватом характеризуется возрастанием количества своих представителей в последующих популяциях. Подобный рост имеет показательный

характер, что следует из выражения (4.2). Для больших популяций эту формулу можно заменить рекуррентной зависимостью вида [176]

$$c(S,k+1)=c(S,k)\frac{F(S,k)}{\bar{F}(k)}. \qquad (4.6)$$

Если допустить, что схема S имеет приспособленность на ε% выше средней, т.е. $F(S,k)=\bar{F}(k)+\varepsilon\bar{F}(k)$, то при подстановке выражения (4.5) в неравенство (4.4) в предположении, что ε не изменяется во времени, при старте от $k=0$ получаем

$$c(S,k)=c(S,0)(1+\varepsilon)^k,$$

$$\varepsilon=\left(F(S,k)-\bar{F}(k)\right)/\bar{F}(k), \qquad (4.7)$$

т.е. $\varepsilon>0$ для схемы с приспособленностью выше средней и $\varepsilon<0$ - в противном случае.

Равенство (4.7) описывает геометрическую прогрессию. Из этого следует, что в процессе репродукции схемы, оказавшиеся лучше (хуже) средних, выбираются на очередных итерациях генетического алгоритма в показательно возрастающих (убывающих) количествах. Обратим внимание, что зависимости (4.2.)-(4.6) основаны на предложении, что функция приспособленности F принимает только положительные значения. При использовании генетических алгоритмов для решения оптимизационных задач, в которых целевая функция может принимать и отрицательные значения, необходимы некоторые дополнительные соотношения между оптимизируемой функцией и функцией приспособленности. Конечный результат, получаемый из выражений (4.3)-(4.5), можно сформулировать в форме теоремы. Это основная теорема генетических алгоритмов, иначе называемая теоремой о схемах [175].

Схемы малого порядка, с малым охватом и с приспособленностью выше средней формирует показательно возрастающее количество своих представителей в последующих поколениях генетического алгоритма.

В соответствии с приведенной теоремой важным вопросом становиться кодирование, которое должно обеспечивать построение схем малого порядка, с малым охватом и с приспособленностью выше средней.

Для эволюционных алгоритмов известно[171], что сходимость к глобальному оптимуму в задаче оптимизации достигается в том случае, если есть уверенность в том, что алгоритм находит решения

за конечное число шагов, и если такое решение будет оставаться в дальнейшем в популяции. Поскольку состояния переходов эволюционных алгоритмов имеют стохастический характер, детерминированная концепция сходимости нее может быть использована для определения срока действия таких алгоритмов. Существуют две широко используемые методы стохастической сходимости эволюционных алгоритмов – это полное совпадения и совпадение по значению [171,173].

Определение 4.1. Пусть x есть случайная переменная и $x_t, t > 0$ - последовательность случайных переменных. Тогда последовательность x_t сходиться полностью в x для любых $\varepsilon > 0$, если $\lim_{t\to\infty} \sum_{i=0}^{t} P\left(\left|x_i - x\right| > \varepsilon\right) < \infty$.

Определение 4.2. Пусть x есть случайная переменная и $x_t, t > 0$ - последовательность случайных переменных. Тогда последовательность x_t будет совпадать по значению в x, если

$$\lim_{t\to\infty} E\left[\left|x_i - x\right|\right] = 0.$$

Приведем дополнительно определения для сходимости генетического алгоритма:

Определение 4.3. Пусть $x_t : t \geq 0$ является последовательностью популяций хромосом, сгенерированной генетическим алгоритмом, и пусть F_t является соответствующим значением лучшего хромосомы в популяции в момент времени t. Генетический алгоритм имеет полную сходимость к глобальному оптимуму f^* задачи оптимизации, определенной функцией $f : x \to R$, если неотрицательная случайная последовательность $D_t = f^* - F_t$ сходиться полностью в 0.

Определение 4.4. Пусть $x_t : t \geq 0$ является последовательностью популяций хромосом, сгенерированной генетическим алгоритмом и пусть F_t является соответствующим значением лучшего хромосомы в популяции в момент времени t. Генетические алгоритмы имеют сходимость по значению к глобальному оптимуму f^* задачи оптимизации, определенной функцией $f : x \to R$, если неотрицательная случайная последовательность $D_t = f^* - F_t$ сходиться полностью в 0.

В общем виде один шаг работы генетического алгоритма можно представить следующим образом:

$$(x'_1,...,x'_n) = sel(x_1,...,x_n);$$
$$(x''_1,...,x''_n) = cross(x'_1,...,x'_n);$$
$$(x'''_1,...,x'''_n) = mut(x'_1,...,x'_n);$$

где $(x_1,...,x_n) \in x^n$ - текущая популяция хромосом; $(x'_1,...,x'_n)$ - популяция хромосом, возникающая в результате селекции; $(x''_1,...,x''_n)$- популяция хромосом, возникающая в результате скрещивания; $(x'''_1,...,x'''_n)$- популяция хромосом, возникающая в результате мутации.

В [172] доказано, что эволюционные алгоритмы сходиться как в среднем, так и полностью к глобальному оптимуму при выполнении следующих условий:

1. Каждая хромосома в популяции может быть изменена на произвольную другую хромосому в одной единственной мутации с вероятностью $p > 0$.
2. Лучшая хромосома в популяции выживает в каждом поколении с вероятностью $p = 1$.

Формально эти условия можно представить в виде:

$$\forall x, y \in X \qquad p\{y = mut(x)\} \geq \delta_m > 0;$$
$$p\{V_n^*(sel(x_1,...,x_k)) = V_k^*(x_1,...,x_k)\} = 1,$$

где V_i^* оператор возврата лучшей хромосомы из i хромосом популяции.

Если условие 1 действительно, можно показать, что эволюционный алгоритм сходиться к глобальному оптимуму за конечное число шагов с вероятностью $p = 1$, независимо от его инициализации, но может сходиться, если нельзя гарантировать, что оптимум действительно останется в популяции после того, как он был найден. Если условие 2 тоже верно, то можно показать, что эволюционный алгоритм сходиться к глобальному оптимуму.

В генетическом алгоритме оператор скрещивания тоже принимает участие в механизме выбора эволюционного процесса. Следовательно, условие 1 может быть применено к генетическому алгоритму в отношении мутации и скрещивания, а оператор селекции нужно рассматривать в формальном описании условия 2.

Тогда выражения

$$p\{V_n^*(sel(x_1,...,x_n)) = V_k^*(x'_1,...,x'_n)\} = 1$$
$$p\{V_n^*(cross(x'_1,...,x'_n)) = V_k^*(x''_1,...,x''_n)\} = 1$$

правильно описывают условие 2 по отношению к ГА.

Покажем, что генетический алгоритм сходиться полностью и в среднем к глобальному оптимуму. Для этого покажем, что условия 1 и 2 удовлетворяются. Операторы мутации и скрещивания могут вносит изменения в хромосомы, использования которых способствует поиску оптимума.

Пусть битовые строки хромосом длиною L представлены вектором $\{0,1\}^L$. Если при сравнении цепочка бит длиною $(L-c)$ в хромосоме представляет оптимум, следовательно, в с битах хромосом и не совпадет с оптимум. Тогда вероятность оператора мутации и скрещивания для достижения глобального оптимума за один шаг равна $P_c^{(L)} = \frac{c!}{L_c} \cdot \frac{1}{L}$. Здесь благоприятное число выборов $c!$ является числом перестановок с элементов из возможного числа выборов L_c. Эта вероятность должна быть умножена на вероятность того, что оператор мутации случайном образом изменяет биты, т.е. на $1/L$, которая является вероятностью того, что $c = r$, где r-случайно выбранная число битов, которые подвергаются мутации.

Если хромосома представлено вектором $\{0,1,...,k-1\}^L$, каждый элемент которого взят из алфавита из k элементов, то вероятность того, что каждая из цифр, которые будут подвержены мутации и скрещиванию, будет являться цифрой оптимальной цепочки, равна

$$P_c^{(L)} = \frac{c!}{L_c} \cdot \frac{1}{(k-1)^c} \cdot \frac{1}{L}.$$

Если $P_c^{(L)}$ всегда положительно, то условие 1 выполняется. При этом следует отметить, что существует также вероятность $p_s > 0$ того, что оптимум случайно введен в популяцию оператором селекции, который нужно рассматривать, хотя рассмотрена вероятностном $P_c^{(L)}$ достаточно, чтобы показать выполнения условия 1.

Чтобы показать, что условие 2 также выполняется, необходимо учесть все операторы, действующие на популяцию, а также показать, что ни один из них не приводит к потере оптимального решения, если оно будет найдено. Проанализируем операторы генетического алгоритма.

Оператор селекции, после избавления от наименее пригодных хромосом, пополняет популяцию новыми хромосомами, поэтому оптимум не может бить потерян.

Оператор мутации и скрещивания действуют только на хромосомы популяции, не трогая при этом лучшую хромосому в популяции. Следовательно, эти операторы также не теряют оптимум. Таким образом, приведенные рассуждение показывают выполнения условия 2. Следовательно, имеет место сходимости генетического алгоритма, если операторы мутации, скрещивания и селекции удовлетворяют условию 1, и условие 2 тоже выполняется.

Покажем, что генетический алгоритм может не сходиться полностью к глобальному оптимуму независимо от его инициализации, если оператор мутации меняет отдельные биты и используется не элитный вариант оператора скрещивания.

Выше было показано выполнение условия 1, в соответствии с которым генетический алгоритм достигает оптимума при условии, что он использует операторы мутации, скрещивания и (или) селекции. Необходимо показать выполнение условия 2, которое является необходимом, но не достаточным условием сходимости.

Чтобы показать, что генетический алгоритм не сходиться полностью к глобальному оптимуму, достаточно показать, что каждый раз, когда найден оптимум, будет существовать следующие поколение хромосом, котором генетический алгоритм не имеет оптимума в популяции. Если есть вероятность, даже очень малая, того, что генетический алгоритм выйдет из оптимума, найдя в другой момент времени другое оптимальное решение, то это значит, что генетический алгоритм не сходиться полностью с вероятностью 1. Это вероятность гарантируются также неэлитизмом оператора селекции в том смысле, что хромосома, представляющее оптимизм при достижением возраста через t поколений, будет изменены в популяции.

Оператор селекции создаст популяцию, состоящий из копий оптимальных решений x_1 и из неоптимальных решений x_2. Так как оператор мутации всегда изменяет, по крайней мере, один бит, хромосомы оптимальных решений не будут достигать глобальных оптимумов. С другой стороны, каждое неоптимальное решение может стать оптимальным в следующем шаге с вероятностью p такой, что $p \le p_{d=1}$, где $p_{d=1}$ является вероятностью достижения оптимума из наиболее вероятного состояния с шагом $d = 1$.

Эти рассуждения доказывают следующее утверждение 4.1.

Утверждение 4.1. Анализ сходимости генетического алгоритма основан на выполнении двух условий:

- В результате мутации и скрещивания можно достичь оптимальное состояние из неоптимального за один шаг;
- Как только оптимальное состояние будет найдено, оно сохранится в популяции и не будет утеряно.

С использованием генетического алгоритма можно решить задачи оптимизации, классификации на основе нечеткого логического вывода и обучение.

4.3. Применение генетических алгоритмов к решению задач нахождения глобального оптимума

4.3.1. Решение Диофантова уравнения

Рассмотрим Диофантово (только целые решения) уравнение: $a+2b+3c+4d=30$, где a, b, c и d - некоторые положительные целые. Применение ГА за очень короткое время находит искомое решение (a, b, c, d).

Конечно, Вы можете спросить: почему бы не использовать метод грубой силы: просто не подставить все возможные значения a, b, c, d (очевидно, 1 <= a,b,c,d <= 30)?

Архитектура ГА-систем позволяет найти решение быстрее за счет более «осмысленного» перебора. Мы не перебираем все подряд, но приближаемся от случайно выбранных решений к лучшим.

Для начала выберем 5 случайных решений: 1 =< a,b,c,d =< 30. Вообще говоря, мы можем использовать меньшее ограничение для b,c,d, но для упрощения пусть будет 30.

Таблица 4.1

1-е поколение хромосом и их содержимое

Хромосома	(a,b,c,d)
1	(1,28,15,3)
2	(14,9,2,4)
3	(13,5,7,3)
4	(23,8,16,19)
5	(9,13,5,2)

Чтобы вычислить коэффициенты выживаемости (fitness), подставим каждое решение в выражение $a+2b+3c+4d$. Расстояние от полученного значения до 30 и будет нужным значением.

Таблица 4.2

Коэффициенты выживаемости первого поколения хромосом (набора решений)

Хромосома	Коэффициент выживаемости
1	|114-30|=84
2	|54-30|=24
3	|56-30|=26
4	|163-30|=133
5	|58-30|=28

Так как меньшие значения ближе к 30, то они более желательны. В нашем случае большие численные значения коэффициентов выживаемости подходят, увы, меньше. Чтобы создать систему, где хромосомы с более подходящими значениями имеют большие шансы оказаться родителями, мы должны вычислить, с какой вероятностью (в %) может быть выбрана каждая. Одно решение заключается в том, чтобы взять сумму обратных значений коэффициентов, и исходя из этого вычислять проценты. (Заметим, что все решения были сгенерированы Генератором Случайных Чисел - ГСЧ)

Таблица 4.3

Вероятность оказаться родителем

Хромосома	Подходимость
1	(1/84)/0.135266 = 8.80%
2	(1/24)/0.135266 = 30.8%
3	(1/26)/0.135266 = 28.4%
4	(1/133)/0.135266 = 5.56%
5	(1/28)/0.135266 = 26.4%

Для выбора 5-и пар родителей (каждая из которых будет иметь 1 потомка, всего - 5 новых решений), представим, что у нас есть 10000-стонняя игральная кость, на 880 сторонах отмечена хромосома 1, на 3080 - хромосома 2, на 2640 сторонах - хромосома 3, на 556 - хромосома 4 и на 2640 сторонах отмечена хромосома 5. Чтобы

выбрать первую пару, кидаем кость два раза и выбираем выпавшие хромосомы. Таким же образом выбирая остальных, получаем:

Таблица 4.4

Симуляция выбора родителей

Хромосома отца	Хромосома матери
3	1
5	2
3	5
2	5
5	3

Каждый потомок содержит информацию о генах и отца и от матери. Вообще говоря, это можно обеспечить различными способами, однако в нашем случае можно использовать т.н. "кроссовер" (cross-over). Пусть мать содержит следующий набор решений: a1,b1,c1,d1, а отец - a2,b2,c2,d2, тогда возможно 6 различных кросс-оверов (| = разделительная линия):

Таблица 4.5

Кроссоверы между родителями

Хромосома-отец	Хромосома-мать	Хромосома-потомок
a1 | b1,c1,d1	a2 | b2,c2,d2	a1,b2,c2,d2 or a2,b1,c1,d1
a1,b1 | c1,d1	a2,b2 | c2,d2	a1,b1,c2,d2 or a2,b2,c1,d1
a1,b1,c1 | d1	a2,b2,c2 | d2	a1,b1,c1,d2 or a2,b2,c2,d1

Есть достаточно много путей передачи информации потомку, и кроссовер - только один из них. Расположение разделителя может быть абсолютно произвольным, как и то, отец или мать будут слева от черты.

А теперь попробуем проделать это с нашими потомками

Таблица 4.6

Симуляция кросс-оверов хромосом родителей

Хромосома-отец	Хромосома-мать	Хромосома-потомок
(13 | 5,7,3)	(1 | 28,15,3)	(13,28,15,3)
(9,13 | 5,2)	(14,9 | 2,4)	(9,13,2,4)
(13,5,7 | 3)	(9,13,5 | 2)	(13,5,7,2)
(14 | 9,2,4)	(9 | 13,5,2)	(14,13,5,2)
(13,5 | 7, 3)	(9,13 | 5, 2)	(13,5,5,2)

Теперь мы можем вычислить коэффициенты выживаемости (fitness) потомков.

Таблица 4.7

Коэффициенты выживаемости потомков (fitness)

Хромосома-потомок	Коэффициент выживаемости
(13,28,15,3)	|126-30|=96
(9,13,2,4)	|57-30|=27
(13,5,7,2)	|57-30|=22
(14,13,5,2)	|63-30|=33
(13,5,5,2)	|46-30|=16

Средняя приспособленность (fitness) потомков оказалась 38.8, в то время как у родителей этот коэффициент равнялся 59.4. Следующее поколение может мутировать. Например, мы можем заменить одно из значений какой-нибудь хромосомы на случайное целое от 1 до 30. Продолжая, таким образом, одна хромосома, в конце концов, достигнет коэффициента выживаемости 0, то есть станет решением. Системы с большей популяцией (например, 50 вместо 5-и) сходятся к желаемому уровню (0) более быстро и стабильно.

4.3.2. Достоинства и недостатки стандартных и генетических методов на примере классической задачи коммивояжера

Генетический алгоритм - новейший, но не единственно возможный способ решения задач оптимизации. С давних пор известны два основных пути решения таких задач - переборный и локально-градиентный. У этих методов свои достоинства и недостатки, и в каждом конкретном случае следует подумать, какой из них выбрать.

Рассмотрим достоинства и недостатки стандартных и генетических методов на примере классической задачи коммивояжера (TSP - travelling salesman problem). [119.177] Суть задачи состоит в том, чтобы найти кратчайший замкнутый путь обхода нескольких городов, заданных своими координатами (рис. 4.1). Оказывается, что уже для 30 городов поиск оптимального пути представляет собой сложную задачу, побудившую развитие различных новых методов (в том числе нейросетей и генетических алгоритмов).

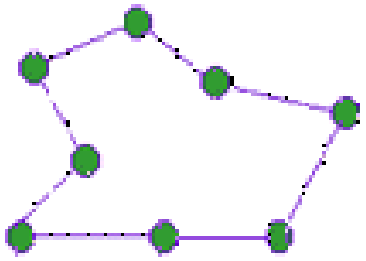


Рис. 4.1. Кратчайший путь

Каждый вариант решения (для 30 городов) - это числовая строка, где на *j*-ом месте стоит номер *j*-ого по порядку обхода города. Таким образом, в этой задаче 30 параметров, причем не все комбинации значений допустимы. Естественно, первой идеей является полный перебор всех вариантов обхода (рис. 4.3.2).

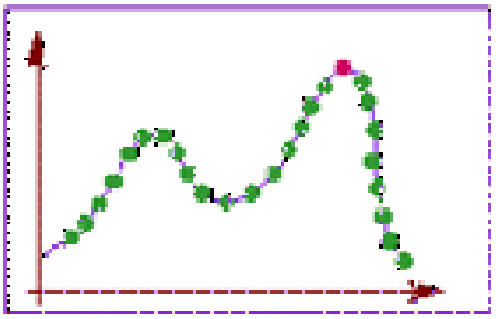


Рис.4.2. Переборный метод

Переборный метод наиболее прост по своей сути и тривиален в программировании. Для поиска оптимального решения (точки максимума целевой функции) требуется последовательно вычислить значения целевой функции во всех возможных точках, запоминая максимальное из них.

Недостатком этого метода является большая вычислительная стоимость. В частности, в задаче коммивояжера потребуется просчитать длины более 1030 вариантов путей, что совершенно нереально. Однако, если перебор всех вариантов за разумное время возможен, то можно быть абсолютно уверенным в том, что найденное решение действительно оптимально.

Второй популярный способ основан на методе градиентного спуска (рис. 4.3). При этом вначале выбираются некоторые случайные значения параметров, а затем эти значения постепенно изменяют, добиваясь наибольшей скорости роста целевой функции. Достигнув локального максимума, такой алгоритм останавливается, поэтому для поиска глобального оптимума потребуются дополнительные усилия.

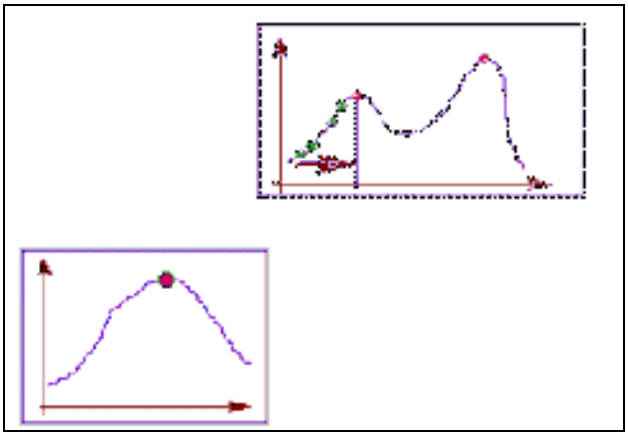


Рис. 4.3. Метод градиентного спуска

Градиентные методы работают очень быстро, но не гарантируют оптимальности найденного решения. Они идеальны для применения в так называемых унимодальных задачах, где целевая функция имеет единственный локальный максимум (он же - глобальный). Легко видеть, что задача коммивояжера унимодальной не является.

Типичная практическая задача, как правило, мультимодальна и многомерна, то есть содержит много параметров. Для таких задач не существует ни одного универсального метода, который позволял бы достаточно быстро найти абсолютно точное решение (рис. 4.4).

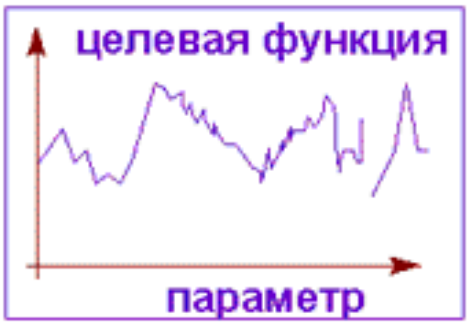


Рис. 4.4. Мультимодальная и многомерная задача

Однако, комбинируя переборный и градиентный методы, можно надеяться получить хотя бы приближенное решение, точность которого будет возрастать при увеличении времени расчета (рис. 4.5) .

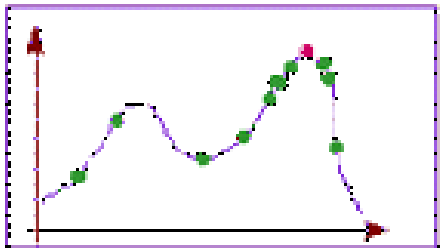


Рис. 4.5. Комбинация переборных и градиентных методов

Генетический алгоритм представляет собой именно такой комбинированный метод (рис. 4.6). Механизмы скрещивания и мутации в каком-то смысле реализуют переборную часть метода, а отбор лучших решений - градиентный спуск. На рисунке показано, что такая комбинация позволяет обеспечить устойчиво хорошую эффективность генетического поиска для любых типов задач.

Рис. 4.6. Генетический алгоритм

Итак, если на некотором множестве задана сложная функция от нескольких переменных, то генетический алгоритм - это программа, которая за разумное время находит точку, где значение функции

достаточно близко к максимально возможному. Выбирая приемлемое время расчета, мы получим одно из лучших решений, которые вообще возможно получить за это время.

4.3.3. Проблема рюкзака

"Проблема рюкзака" (или "ранца") может быть сформулирована следующим образом. Пусть задано множество натуральных чисел $A=(a_1,a_2,...,a_n)$ и натуральное число *S*. Требуется установить, имеется ли такое подмножество множества *A*, сумма элементов которого была бы равна *S*. Эквивалентной является следующая формулировка: существует ли такой набор чисел x_i из (0,1), $i \le n$, для которого $\sum a_i x_i = S \ (1 \le i \le n)$.

Данная проблема получила свое название в связи с тем, что поставленная задача может быть переформулирована также в следующем виде. Имеется набор предметов с известными весами и рюкзак, который может выдержать вес, не превышающий заданной величины. Можно ли выбрать набор предметов для погрузки в рюкзак так, чтобы они в точности имели максимально возможный вес.

"Проблема рюкзака" является весьма сложной, ее решение с полиномиальной сложностью в настоящее время не известно.

Идея построения системы шифрования на основе проблемы рюкзака заключается в выделении некоторого подкласса задач об укладке рюкзака, решаемых сравнительно легко, и "маскировки" задач этого класса (с помощью некоторого преобразования параметров) под общий случай.

Параметры подкласса определяют секретный ключ, а параметры модифицированной задачи – открытый ключ. В качестве легко решаемой задачи Р. Меркль и М. Хеллман в 1978 г. предложили задачу об укладке "супервозрастающего" рюкзака. Изложим ее суть.

Назовем супервозрастающей последовательность натуральных чисел $(b_1,b_2,...,b_n)$, обладающую свойством $b_i > \sum b_j, 1 \le j \le i-1, \ 2 < i < n$. Можно убедиться в том, что проблема рюкзака для супервозрастающей последовательности может быть решен помощью процедуры, состоящей в выполнении следующих шагов:

- Положить $i = n$;
- Если $i > 1$, то положить x_i равным 1 и *S* равным $S - b_i$, если $S > b_i$, и положить x_i равным 0 в противном случае;

- Положить i равным $i-1$ и возвратиться к шагу 2.

В системе, основанной на проблеме рюкзака, величина S является параметром системы.

Для вычисления открытого и соответствующего секретного ключа каждый из абонентов системы осуществляет следующую последовательность действий.

1. Выбирает супервозрастающую последовательность $(b_1, b_2, ..., b_n)$, и модуль m такой, что $m > \sum b_i,\ 1 \le i \le n$.

2. Выбирает случайное число $W, 1 < W < m-1$, такое что НОД(W,m)=1

3. Выбирает случайную перестановку π чисел (1, 2,...,n).

4. Вычисляет $a_i = (W * b\pi(i)) \bmod(n)$ для $i = 1...n$.

Открытым ключом является набор $(a_1, a_2, ..., a_n)$, секретным ключом – набор $(\pi, m, W, (b_1, b_2, ..., b_n))$. Чтобы зашифровать сообщение M, предназначенное для абонента A, абонент B осуществляет следующие шаги с помощью открытого ключа $(a_1, a_2, ..., a_n)$ абонента A:

a. Представляет M в виде бинарной последовательности $M = M_1 M_2 ... M_n$ длины n;

b. Вычисляет $C = \sum M_i * a_i,\ i = 1...n$ и направляет его к A.

Абонент A, получив C, вычисляет $H = (W - 1 * C) \bmod(m)$, а затем, решая проблему рюкзака для супервозрастающей последовательности, находит числа $z_i,\ i = (0,1)$ такие, что $H = \sum z_i * b_i,\ (i = 1,...,n)$. Биты последовательности M_i вычисляются по формуле: $M_i = z\pi(i),\ i = 1,...,n$.

Корректность проведенной процедуры расшифрования вытекает из следующих рассуждений. Поскольку $H = W - 1 * C = W - 1 * \sum M_i * a_i = (\sum M_i * b\pi(i))\ (\bmod m)$ и $0 < H < m$, то $H = \sum M_i * b\pi(i),\ (i = 1,...,n)$, и, следовательно, алгоритм решения проблемы рюкзака действительно находит биты открытого текста, переставленные в соответствии с перестановкой π.

4.3.3.1. Реализация алгоритма

В качестве примера, приведём программу которая шифрует/дешифрует текстовые файлы заменяя один символ другим, вычисляя его по алгоритму на основе "проблемы рюкзака". В данном случае, алгоритм применяется к байтам, т.е. символам текста.

Программа включает в себя универсальный кодер-декодер способный работать с любыми текстовыми файлами, и имеющий

гибкую возможность манипулирования входными параметрами, что делает практически невозможным его использование посторонними лицами с целью дешифрования информации.

В процессе создания программы была использован простой и удобный интерфейс, позволяющий выполнить следующие действия:

1. Кодирование входного файла – задание параметров его шифрования; т. е. секретного ключа $(\pi, m, W, (b_1, b_2, ..., b_n))$. Естественно вы можете не все указанные параметры вводить каждый раз при кодировании, можно задать их статично в теле самой программы, однако не стоит забывать, что чем больше параметров вашего секретного ключа надо знать для кодирования/декодирования тем сложнее злоумышленнику дешифровать ваши данные.

2. Расшифрование файла – задание параметров расшифрования; т. е. тех же самых параметров указанных выше - $(\pi, m, W, (b_1, b_2, ..., b_n))$.

Конечно, можно ещё немного схитрить. Например, при расшифравании запрашивать параметры, которые не требовалось вводить при кодировании (при кодировании они заданы заранее), но это уже дело вашей фантазии.

4.3.3.2. Немного о криптоанализе рюкзачной системы шифрования

Заметим, что предложенная реализация алгоритма шифрует текстовые файлы, заменяя один символ другим. Очевидно, что в данной ситуации возможно успешное примение атаки на основе частотного анализа. Например, по следующему простейшему алгоритму:

1 Подсчёт частоты встречаемости шифрообозначений, а также их некоторых сочетаний (например, по 2, 3, 4 - символа).

2 Выявление шифрообозначений, заменяющих один символ соответствующим ему другим.

3 Выдвижение гипотез о значениях шифрообозначений и их проверка.

Здесь можно использовать практически любую достоверную и не очень информацию об открытом тексте, например если заранее известен формат (XML) или набор используемых символов. Следует уточнить, что реализация алгоритма шифрования поддаётся совершенствованию, и только для наглядности она представлена в данном виде. Приведём несколько советов, как можно это сделать:

- Можно кодировать один символ нескольмими - WORDом например;
- Ничто не мешает кодировать не по одному символу, а сразу несколько символов.
- Приведенные выше методы, к сожалению не избавляют от раскрытия кода на основе статистического анализа. Для предотвращения этого можно добавить любую, хотя бы самую простую, зависимость одного шифруемого символа от другого, например соседнего (или нескольких символов).

Заметим, что вне зависимости от реализации рюкзачный алгоритм шифрования не идеален. Доказано, что существует алгоритм полиномиальной сложности, который может быть использован противником для получения открытого текста M по шифротексту *C*. Пусть противнику известна последовательность $\{a_i\}$, этот алгоритм находит пару таких целых чисел u_1, m_1, что отношение u_1/m_1 близко к отношению u/m (где $u = W - 1 \bmod(m), a, W, m$ являются частью секретного ключа). Кроме того, числа $B_i = (u_i * a_i) \bmod(m), 1 < i < n$, образуют супервозрастающую последовательность. Эта последовательность затем используется противником вместо $(b_1, b_2, ..., b_n)$ для дешифрования сообщения.

Понятие идеальности алгоритма в криптографии вообще очень сложное. В определённых случаях нам нужен DES, RSA или Blow-fish, а где-то подойдёт рюкзачная система шифрования. Не будем забывать о том, что, во-первых, злоумышленник не должен вообще знать о том какая криптосистема применяется для защиты данных, и, во-вторых, он не должен иметь возможности получать в каком либо виде криптотекст и уж тем более доступ к самой программе-кодеру. Но даже в этом случае нельзя быть на 100% уверенным в защите ваших данных. Однако, это лучше чем не предпринимать вообще ничего).

При грамотном администрировании, таком как ограничение доступа к данным (принцип наименьших привилегий), возможность выполнения программы на сервере только тому, кому это разрешено, можно существенно снизить все основные угрозы безопасности ваших данных.

4.3.4. Общие рекомендации к программной реализации генетического алгоритма

Поскольку генетические алгоритмы представляют собой достаточно универсальный подход к решению экстремальных задач, то выбор языка программирования не играет большой роли.

Саму программу можно писать, используя как объектно-ориентированный подход, так и структурный. Ниже предлагается способ реализации различных компонентов генетического алгоритма при использовании обоих подходов (Табл. 4.8).

Таблица 4.8

Вариант реализации компонентов ГА

Компонент генетического алгоритма	Структурный подход	Объектно-ориентированный подход
Особь	Одномерный массив для записи значений генов. Размерность массива совпадает с количеством генов у одной особи (количество генов равно числу настраиваемых параметров)	Класс «Особь», содержащий массив генов.
Популяция	Двумерный массив, в котором i-я строка содержит гены i-й особи.	Отдельный класс «Популяция», содержащий одномерный массив объектов класса, представляющего особь.
Оценивание популяции	Подпрограмма оценки строк массива популяции в соответствии с выбранной целевой	Метод управляющего класса, оценивающий популяцию в соответствии с выбранной целевой

	функцией.	функцией.
Приспособленность популяции	Одномерный массив, в котором *i*-й элемент соответствует приспособленности *i*-й особи.	Одномерный массив со значениями ошибок особей, входящий в управляющий класс.
Особи, выбранные для скрещивания	Двумерный массив, строки которого соответствуют хромосомам особей, выбранным для скрещивания.	Объект класса «Популяция», содержащий объекты класса «Особь», соответствующие выбранным особям
Реализация скрещивания, мутации, формирования нового поколения.	Подпрограммы, обрабатывающие элементы массива, представляющего популяцию особей, а также популяцию особей, выбранных для скрещивания.	Методы управляющего класса, работающие с основной популяцией и популяцией особей для скрещивания.

Приведенный в табл. 4.8 способ реализации генетического алгоритма не является эталонным и, вполне возможно, далек от идеала. Данные в табл. 4.8 могут служить в качестве «опорных» для конкретной реализации генетического алгоритма. Отметим, что большую гибкость и расширяемость программной реализации не только генетического алгоритма, но и любого другого алгоритма и системы вообще, можно достичь, используя компонентно-ориентированный подход и паттерны проектирования.

4.4. Применение нечетких генетических алгоритмов с искусственным отбором для решения задач оптимизации

Эволюционные и генетические алгоритмы успешно применяются для решения задач структурной и параметрической оптимизации, возникающих в ходе управления и проектирования сложных интеллектуальных систем, а также при принятии решений в условиях неопределенной или неполной информации.

В то же время в ходе становления и развития каждого из этих направлений становилось очевидным наличие глубоких взаимосвязей, взаимозависимости между ними. Изучение этих взаимозависимостей, их влияния на конечный результат и в итоге их практическое применение, и является, по-видимому, главной задачей вычислительного интеллекта.

Оказалось, что соединение в единое целое разнородных на первый взгляд составляющих дает синергетический (нелинейный) эффект, который не только превосходит ожидаемый эффект, но и позволяет во многом компенсировать минусы, присущие каждому методу в отдельности [166-168].

Известно, что эволюционные (генетические) алгоритмы давно и успешно применяются для подбора весов и синтеза топологии и архитектуры нейронных сетей. Они также используются для формирования базы правил и функции принадлежности нечетких систем [169].

В свою очередь нейронные сети позволяют подбирать соответствующие значения параметров генетических алгоритмов, и обеспечивают нечетким системам возможность обучения.

И, наконец, методы теории нечетких множеств используются как для подбора параметров генетических алгоритмов, так и для выбора коэффициентов определяющих скорость обучения нейронных сетей [169].

Примерами подобного рода кооперации могут служить нейро-логические, нейро-нечеткие модели, нечеткие логические регуляторы и т.д. [166, 170, 171].

Еще одним интересным моментом является нечеткое кодирование генетических алгоритмов. Известно, что для применения генетических алгоритмов все переменные задачи оптимизации должны быть определенным образом закодированы. Сущность кодирования заключается в преобразовании любого

числового значения в некоторую последовательность символов конечного алфавита, состоящего обычно из небольшого числа элементов. Наиболее известный пример такого кодирования – это двоичное кодирование и представление решений в виде последовательности нулей и единиц.

Очевидно, что можно провести аналогию между процессами кодирования и фаззификации, то есть преобразования исходных числовых величин в распределения, соответствующие термам лингвистической переменной. При этом каждое числовое значение описывается одним или несколькими термами, а степень его соответствия определяется степенью принадлежности нечеткому множеству [166,167].

Термины конечного алфавита нечетких множеств могут включать такие понятия как «больше», «меньше», «немного больше (меньше)», «значительно больше (меньше)», «примерно ноль», «малая отрицательная (положительная)» и др. Использование таких алфавитов при кодировании решений в генетических алгоритмах дает возможность построить неоднородное разделение пространства поиска. Кроме того, закодированные последовательности могут иметь различную степень детализации.

Степень детализации и характер распределения решений в пространстве поиска может определяться на основе начальных знаний о решаемой задаче. Такое кодирование позволяет сосредоточить основные усилия на поиске в наиболее перспективных областях. В свою очередь для поиска в неперспективных областях (не содержащих элементов, которые оптимизируют функцию пригодности), определяют последовательности относительно низкой степени детализации.

Таким образом, можно отметить, по крайней мере, два преимущества нечеткого кодирования. Во-первых, кодовые последовательности могут быть неоднородными и ориентироваться на отдельные многообещающие области поиска, что позволит сократить область поиска и соответственно вычислительные затраты. Кроме того, в самой закодированной последовательности может быть неявным образом включена определенная функция пригодности.

Во-вторых, нечеткое кодирование позволяет выполнять так называемое слабое кодирование оптимизируемых структур. Слабое кодирование, в отличие от обычного (сильного) кодирования, не

подразумевает жесткое соответствие типа «один – к – одному» между генотипом и фенотипом в кодируемой структуре.

Основная идея слабого кодирования состоит в использовании отношений типа «один - ко – многим» в процессе задания соответствия генотип - фенотип. Благодаря этому мы получаем единственный генотип, которому соответствует нечеткое семейство фенотипов. Это позволяет нам использовать все преимущества лингвистических переменных при сохранении двоичного кодирования.

Широкое распространение получили также методики кодирования решений в виде вещественных чисел [172 - 175].

Логическим следствием описанного процесса взаимопроникновения и интеграции различных компонент триады вычислительного интеллекта явилось появление в качестве самостоятельного термина понятия «нечеткие генетические алгоритмы».

4.4.1. Нечеткие генетические алгоритмы

Термин «нечеткие генетические алгоритмы» был впервые введен в обращение в трудах ученых университета Гранады (Испания) Ф. Эррера и М. Лозано [176].

Согласно определению, данному в работе [170] нечеткий генетический алгоритм – это генетический алгоритм в котором некоторые компоненты реализованы с использованием инструментов нечеткой логики. Таких как нечеткие операторы и нечеткие правила для создания генетических операторов с различными свойствами; системы нечеткого логического контроля параметров ГА в соответствии принятыми критериями; нечеткие критерии остановки процесса генетического поиска. Математический аппарат теории нечетких систем используется в данном случае для кодирования, подбора оптимальных параметров генетических алгоритмов, выбора функции пригодности и критерия останова, создания нечетких генетических операторов.

Можно считать, что нечеткие генетические алгоритмы представляют собой частный случай нечетких адаптивных генетических алгоритмов [176, 177], базирующихся на применении адаптивных процедур и правил для настройки параметров, выбора операторов и представления целевой функции. В частности для

подбора значений вероятности операторов кроссинговера и мутации могут использоваться правила работы нечеткого логического контроллера [177].

В качестве примера реализации нечетких генетических операторов можно привести нечеткие операторы кроссинговера и мутации, описанные в работе [178]. Предложенные нечеткие операторы кроссинговера и мутации могут эффективно применяться для решения оптимизационных задач на нечетких графах и гиперграфах.

Нечеткий оператор кроссинговера выполняется на основе логических операций булевой алгебры. Новые решения образуются из уже имеющихся за счет использования логических операций И, ИЛИ, И-НЕ и других. Например, при выполнении оператора кроссинговера на основе операции «И» потомок образуется по правилам выполнения конъюнкции нечетких высказываний. Нечеткий оператор мутации (ОМ) выполняется на основе логической операции инверсии. На некотором шаге выполнения алгоритма случайным образом выбирается хромосома (решение). В этой хромосоме случайным образом выбираются две точки, и ко всем разрядам хромосомы находящимся между выбранными точками применяется операция логической инверсии. Данные нечеткие генетические операторы были успешно использованы для решения задачи нахождения нечетких инвариантов нечетких графов (нечеткие внутренне и внешне устойчивые подмножества, нечеткие ядра и клики и т.д.) которые являются весьма актуальными и распространенными моделями задач дискретной оптимизации [178].

В литературе также описано значительное количество других нечетких генетических операторов: нечеткий кроссинговер, основанный на связях [176]; кроссинговеры с использованием множества родителей и с образованием множества потомков; кроссинговер поиска экстремума [174] и др.

4.4.2. Построение модели нечеткого логического вывода на основе использования генетических алгоритмов с искусственным отбором

Пусть задана выборка нечетких экспериментальных данных (X_r, y_r), $r = \overline{1,M}$; здесь $X_r = (x_{r1}, x_{r2}, ..., x_{rn})$ - входной n-мерный вектор и $y_r = (y_1, y_2, ..., y_M)$ - соответствующий ему выходной вектор.

В общем виде требуется построить модель, основанную на нечетких правилах вывода:

$$\bigcup_{p=1}^{k_j}\left(\bigcap_{i=1}^{n} x_i = a_{i,jp} - \text{с весом } w_{jp}\right) \to y_j = b_{m0} + b_{m1}x_1^j + ... + b_{mn}x_n^j.$$

В процессе построения модели нужно найти такие значения коэффициентов правил

$$B = (b_{ij}), i = \overline{1,m},\ j = \overline{0,n},$$

при которых достигается минимум следующего выражения:

$$\sum_{r=1}^{M}\left(y_r - y_r^f\right) \to \min, \qquad (4.8)$$

где y_r^f - результат нечетких правил вывода с параметром B в r-й строке выборки (X_r).

Входной матрице X_r соответствует следующий результат нечеткого вывода:

$$y_r^f = \frac{\sum_{i=1}^{m}\mu_{d_i}(X_r)\cdot d_i}{\sum_{i=1}^{m}\mu_{d_i}(X_r)}; \qquad (4.8)$$

здесь $d_i = b_{i0} + b_{i1}x_{r1} + b_{i2}x_{r2} + ... + b_{in}x_{rn}$ - выход i-правила; $\mu_{d_i}(X_r)$ - функция принадлежности, соответствующая каждой экспериментальной информации:

$$\mu_{d_j}(X_r) = \mu_{i1}(x_{r1})\cdot\mu_{i1}(x_{r2})\cdot\mu_{i1}(x_{r3})\cdot...\cdot\mu_{i1}(x_{rn}) \vee$$
$$\vee\ \mu_{i2}(x_{r1})\cdot\mu_{i2}(x_{r2})\cdot\mu_{i2}(x_{r3})\cdot...\cdot\mu_{i2}(x_{rn}) \vee$$
$$.....$$
$$\vee\ \mu_{im}(x_{r1})\cdot\mu_{im}(x_{r2})\cdot\mu_{im}(x_{r3})\cdot...\cdot\mu_{im}(x_{rn}),$$

Для решения задачи (4.8) используем генетический алгоритм с искусственным отбором [168].

На основе предлагаемого алгоритма лежит синтез обычного эволюционного генетического подхода с идеями адаптационной оптимизации [173] и, прежде всего, последовательного комплекс-

метода отыскания экстремума функций многих переменных [173-175]. При этом в каждый момент времени текущая популяция отождествляется с популяцией - комплексом точек в пространстве поиска, а кроме традиционных генетических операторов мутации, скрещивания и селекции дополнительно вводятся операторы комплекса-поиска такие, как выбор, отражение, растяжение и сжатие. При этом в отличие от традиционного комплекс-метода предлагается отражать не одну наихудшую вершину комплекса, а целое множество наихудших особей популяции.

В общем случае процедура оптимизации на основе обычного последовательного комплекса-метода выглядит следующим образом: требуется отыскать минимум некоторой функции

$$E(x) = \sum_{r=1}^{M} \left(y_r - y_r^f \right)^2 \to \min$$

достаточно общего вида, при этом о характере этой функции не делается практически никаких априорных предложений. Работа алгоритма начинается с формирования начального комплекса

$$x_i(0) = \begin{pmatrix} x_1(0) \\ x_2(0) \\ \vdots \\ x_N(0) \end{pmatrix} = \begin{pmatrix} x_{11}(0) & x_{12}(0) & \dots & x_{1n}(0) \\ x_{21}(0) & x_{22}(0) & \dots & x_{2n}(0) \\ \vdots & & & \\ x_{N1}(0) & x_{N2}(0) & \dots & x_{Nn}(0) \end{pmatrix}, \quad i = \overline{1, N} \geq n+1,$$

представляющего собой популяцию хромосом, достаточно произвольно расположенных в n-мерном пространстве поиска. С начало выполняется операции селекции, затем скрещивания и мутация. От этого получается новая популяция хромосом $x_i(1)$.

После этого производится операция выбора. На этом этапе вычисляется значение функции во всех хромосом и находится средняя годность популяции

$$E_{сред} = \frac{1}{N} \sum_{i=1}^{N} E(x_i(k)).$$

Затем хромосомы с годностью меньше средней по всей популяции заменяется на «наилучшей» хромосому.

Если $E_{сред} < E(x_i(k))$ то будет $x_i(k+1) = x_i(k)$, который дает $\min\limits_{i=1,N} (E(x_i(k)))$. Среди множества этих хромосом находится «наихудшая» $x_i(1)$, в которой значение функции $E(x_H(1))$ максимально, после чего эта точка отражается через центр тяжести всех остальных

вершин-точек, формируя новый комплекс $x_i(1),\ i=\overline{1,N}$. Такое отражение вместе с растяжением и сжатием обеспечивают движение комплекса к экстремуму функции $E(x)$, при этом, благодаря достаточно случайному распределению хромосом в популяции, поиск имеет глобальный характер.

С формальной точки зрения рассмотрим процесс оптимизации на k-й итерации поиска, когда сформирован комплекс $x_i(k),\ i=\overline{1,N}$. Среди множества $x_i(k)$ находится «наихудшая» такая, что

$$E(x_H(k))=\min_i\{E(x_1(k)),...,E(x_H(k))\},$$

после чего определяется центр тяжести популяции без наихудшей точки:

$$x_{cj}(k)=\left(x_{1j}(k)+x_{2j}(k)+...+x_{Nj}(k)-x_{Hj}(k)\right)/(N-1)\qquad j=\overline{1,n}.$$

Далее $x_H(k)$ отражается через центр тяжести $x_c(k)$, формируя новую вершину комплекса $x_R(k)$, которая теоретически расположена ближе к экстремуму, чем $x_H(k)$ и $x_c(k)$, т.е. $E(x_R(k))<E(x_c(k))<E(x_H(k))$.

Операция отражения формально имеет следующий вид:

$$x_R(k)=x_c(k)+\eta_R(x_c(k)-x_H(k))=$$

$$\frac{1}{N-1}x_1(k)+...+\frac{1}{N-1}x_{N-1}(k)+\frac{\eta_R}{N-1}x_1(k)+...$$

$$+\frac{\eta_R}{N-1}x_{N-1}(k)-\eta_R x_H(k)=X(k)R,$$

где η_R - параметр шага отражения, часто полагаемый равным единице, $X(k)=(x_H(k),x_1(k),...,x_{N-1}(k))$ - $(n\times N)$-матрица координат вершин комплекса, $R=\left(-\eta_R,\frac{1+\eta_R}{N-1},...,\frac{1+\eta_R}{N-1}\right)^T$ - $(N\times 1)$-вектор.

В случае, если отражения вершина $x_R(k)$ окажется «наилучшей» среди всех остальных популяции хромосом, т.е. $E(x_R(k))<E(x_i(k))<E(x_H(k)),\quad i=1,2,...,N-1$

производится операция растяжения комплекса в направлении от центра тяжести $x_C(k)$ до $x_R(k)$ согласно выражению $x_E(k)=x_c(k)+\eta_E(x_R(k)-x_c(k))=X(k)E,$ где η_E – параметр шага растяжения, часто полагаемый равным двум,

$$E=\left(-\eta_E,\eta_R,\frac{1-\eta_E(1-\eta_R)}{N-1},...,\frac{1-\eta_E(1-\eta_R)}{N-1}\right)^T.$$

Если же $x_R(k)$ окажется наихудшей среди всех $x_i(k)$, комплекс сжимается согласно соотношению

$$x_S(k) = x_c(k) + \eta S(x_R(k) - x_c(k)) = X(k)S,$$

где η_S – параметр шага сжатия, обычно полагаемый равным 0,5,

$$S = \left(-\eta_S, \eta_R, \frac{1-\eta_S(1-\eta_R)}{N-1}, ..., \frac{1-\eta_S(1-\eta_R)}{N-1} \right)^T.$$

Таким образом, в процессе своего движения к экстремуму оптимизируемой функции комплекс на каждой итерации теряет одну наихудшую вершину и приобретает одну новую точку так, что на $(k+1)$-й итерации новый комплекс также имеет N точек-вершин.

С помощью предложенного подхода решена задача создания модели нечеткого логического вывода на основе использования генетического алгоритма с искусственным отбором и создана программное обеспечение.

Работа алгоритма образована последовательностью следующих шагов:

1. создание начальной популяции, образованной $P(0)$ особями хромосомами – вершинами комплекса;
2. операция скрещивание с увеличением популяции $P_{CR}(0) > P(0)$;
3. операция мутации $P_M(0) > P_{CR}(0)$;
4. первая селекция (определение наихудших особей) без сокращения популяции $P_{SEL1}(0) = P_M(0)$;
5. операция выбора заменяем значения наилучшему во всей популяции;
6. операция отражения с удалением P наихудших особей $P_M(0) < P_{SEL1}(0)$;
7. операция растяжения без увеличения популяции $P_E(0) = P_R(0)$;
8. операция сжатия без увеличения популяции $P_I(0) = P_E(0)$;
9. вторая селекция с удалением $P_W(0)$ наихудших особей $P_{SEL2}(0) = P_I(0) - P_W(0) = P(1)$ и формирование популяции $P(1)$ следующей итерации алгоритма.

Эффективность ГА зависит от выбора способов кроссинговера, мутации, выбора родительской пары, формирования новой популяции, а также от таких параметров как размер популяции, длина хромосомы. Оценить ГА можно с помощью разнообразных тестовых

функций. Все тестовые функции могут иметь различное число параметров (n). Поэтому имеет смысл запустить алгоритм для оптимизации некоторой функции сначала с небольшим n (например,10 или 20), а затем с =50, 100, 200,... Это даст возможность проверить «масштабируемость» алгоритма. Так как генетические алгоритмы используют стохастичность, то для того, чтобы определить, насколько эффективен ГА, нужно запустить его на одной и той же тестовой функции несколько раз, а потом взять средний результат.

Для задачи максимизации функции двух переменных z(x,y)=exp(−x2− y2)+sin(x + y) результаты получились следующие (рис.4.7): максимум функции достигается в точке x =0.462, y =0.462 и f(0.462;0.462)=-1.45055.

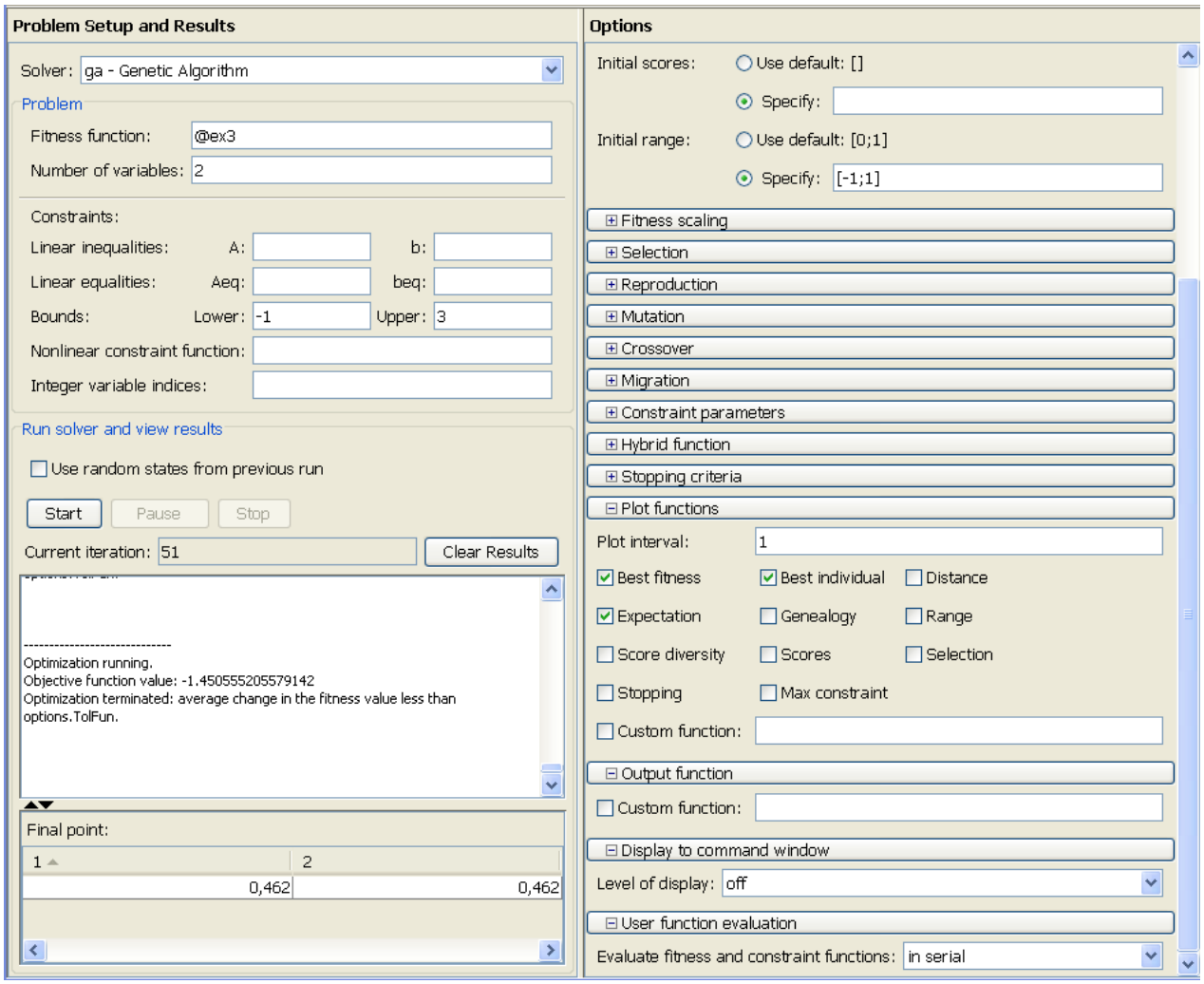


Рис.4.7. Решение задачи максимизации функции двух переменных

Таким образом, применение новых подходов на основе эволюционных методов для оптимизации нечетких баз правил позволит значительно снизить время формирования решения и повысить достоверность его принятия в интеллектуальных системах.

4.5. Применение искусственных иммунных систем к решению задачи оценки состояния сложных процессов

В настоящее время проблема моделирования биологических принципов обработки информации является одной из основных задач искусственного интеллекта. Разработка нетрадиционных биологических подходов при решении широкого круга проблем, начиная от защиты информации и заканчивая проблемами прогнозирования динамических свойств нелинейных объектов управления в реальном масштабе времени, являются весьма актуальными задачами во всем мире. Белки играют огромную роль в природе. Необходим тщательный анализ и осмысление процессов обработки информации белками в естественных системах, что позволит глубже понять механизмы функционирования информационных систем и покажет дальнейшие пути развития данной области науки.

Биологическая иммунная система (ИС) представляет собой сложную распределенную адаптивную систему интеллектуальной обработки информации. ИС защищает организм от инородных вирусов и инфекций. Иммунная система способна к обучению, обладает памятью, решает задачи поиска и классификации. ИС обрабатывает огромные объемы информации, поэтому алгоритмы, созданные природой, оказались эффективными и в математических задачах поиска и распознавания.

Белки это биополимеры сложного строения [179], макромолекулы которых представляют собой остатки аминокислот или полипептидных цепей, соединенных между собой пептидной связью. Именно пространственная структура белка определяет химические, биологические и функциональные свойства белка. Свойства белка могут сильно изменяться при замене хотя бы одной аминокислоты. Это связано с тем, что изменение конфигурации пептидных цепей ведет к другим условиям образования пространственной структуры белка, которая определяет все его функции в организме. Решающая роль в белках принадлежит не отдельным аминокислотным остаткам, а их сочетаниям. В процессе

эволюции сформировались механизмы отбора белковых структур, которые способны сворачиваться в определенные трехмерные нативные структуры, чья энергия существенно ниже, чем энергия альтернативных структур. Основной чертой белковых последовательностей, определяющей их физические свойства, является повышенная стабильность нативной (функциональной) структуры и существование большой щели между энергией нативной структуры и минимальной энергией неверно свернутых структур.

Приведенные свойства естественных белковых структур служат основой биологического подхода искусственных иммунных систем (ИИС) [180]. Основная идея заключается во взаимодействии между белками иммунной системы человека и чужеродными антигенами, то есть в возможности произвольного связывания (так называемого молекулярного узнавания) посредством определения минимальной энергии связи между формальными пептидами.

4.5.1. Проблемы оценивания состояния сложных систем на основе иммунокомпьютинга

Новая вычислительная процедура, называемая искусственные иммунные системы, основанная на принципах иммунной системы, обладает способностью обучаться новой информации, запоминать ранее полученную информацию и осуществлять распознавание образов и анализ данных на основе принципов биомолекулярного узнавания в высоко распределенной манере. Эти системы предлагают мощные и робастные возможности обработки больших массивов информации для решения сложных задач. Строгий математический базис ИИС основан на биологическом прототипе иммунной сети и понятиях формального протеина и формальной иммунной сети (ФИС).

Эти математические модели были названы формальной иммунной системой или иммунокомпьютингом на основе свойств сингулярного разложения произвольных матриц.

Отдельные статьи по ИИС начали появляться в 1980-х годах, однако в отдельно направление ИИС выделились только в середине 90-х годов с появлением работ Форреста, Дасгупты, Ханта и Кука. Первая книга про искусственные иммунные системы вышла в 1998 году под редакцией Дипанкара Дасгупты [181].

Чужеродные агенты, находясь в организме, производят молекулы, называемые антигенами. Большая часть антигеном может

быть распознана специальными клетками – В-лимфоцитами, которые циркулируют в кровеносной и лимфатической системах в ожидании столкновения с антигенами. После того как антиген взаимодействует с антителами В-лимфоцита, стимулируется процесс клонирования лимфоцита. Этот процесс называется клональным отбором. Процесс клонирования В-лимфоцитов в результате взаимодействия с антигенами называется иммунным ответом [182].

Предварительные операции включают в себя создание баз данных на основе статистических временных рядов, характеризующих рассматриваемую систему; создание баз знаний на основе мнений экспертов; нормирование входных параметров; удаление пробелов в таблицах данных; операции по считыванию информации из баз данных, баз знаний и т.д.

Основные операции заключаются в создании иммунной сети, обучении иммунной сети с учителем и.т.д.

Вспомогательные операции состоят из таких как: выделение информационно - ценных признаков и снижение размерности анализируемого пространства признаков; создание оптимальной структуры иммунной сети; тестирование; оценка энергетических погрешностей ИИС при распознавании образов и т.д.

База знаний содержит сведения, которые отражают закономерности, существующие в рассматриваемой предметной области, позволяют выводить новые знания и прогнозировать потенциально возможные состояния исследуемой области; сведения о структуре и содержании базы данных; сведения по языку общения; метазнания, определяющие способы представления и переработки знаний. В базу знаний помещаются как общедоступные данные, так и знания эксперта в данной предметной области, вычислительные алгоритмы реализации процедур иммунокомпьютинга, результаты группировки и автоматической классификации, а также интерпретация результатов вычислений. В вычислительных процедурах иммунокомпьютинга в качестве аналога расстояния используется понятие энергии связи, основанное на сингулярном разложении матрицы. Энергия связи между объектами A и M представляется следующим образом:

$$\omega_i = -U_i^T M V_i,\ U_i^T U_i = 1,\ V_i^T V_i = 1,\ i = 1,2,...,r,$$

где U_i, V_i – соответственно, правые и левые сингулярные векторы матрицы A, r –ранг матрицы.

Алгоритм вычислительной процедуры обучения с экспертом состоит из следующих шагов:

Шаг 1. Сворачивание вектора в матрицу. Заданный вектор *X* размерности $(n \times 1)$ сворачиваем в матрицу *M* размерности $n_U \times n_V = n$.

Шаг 2. Формируем матрицы A_1, $A_2, \ldots, A_k$ для эталонных классов $c = 1, \ldots, к$ и вычисляем их сингулярные векторы: $\{U_1, V_1\}$ – для A_1, $\{U_2, V_2\}$ - для A_2, $\{U_k, V_k\}$ - для A_k.

Шаг 3. Распознавание. Для каждого входного образа *M* вычисляем *к* значений энергии связи между каждой парой сингулярных векторов:

$$\omega_1 = -U_1^T M V_1, \ \ldots\ldots, \ \omega_k = -U_k^T M V_k .$$

Шаг 4. Определяем класс, к которому принадлежит входной образ *M.* Минимальное значение энергии связи ω^* определяет этот класс:

$$c = \omega^* = \min_c \{\omega_c\} .$$

4.5.2. Методы оценки состояния слабоформализуемых процессов на основе использования иммунных алгоритмов

Основные вычислительные процедуры оценки состояния слабоформализуемых процессов основаны на свойствах сингулярного разложения произвольных матриц и математическом аппарате иммунокомпьютинга.

Как известно, для произвольной матрицы *A* размерности $(m \times n)$ существует так называемое сингулярное разложение, т.е. представление матрицы в виде

$$A = USV^T , \qquad (4.9)$$

где $U - (m \times m)$ и $V - (n \times n) -$ ортогональные квадратные матрицы, удовлетворяющие критерию ортогональности:

$$VV^T = V^T V = E_{m \times m} , \qquad UU^T = U^T U = E_{n \times n} ,$$

где *E* - единичные матрицы соответствующих размерностей.

Матрица *S* состоит из квадратного диагонального блока размерности $r \times r$ $(r = \min(m,n))$ с неотрицательными элементами на главной диагонали и, если $n \neq m$, из дополнительных нулевых строк или столбцов

$$S = [S^{'};0], \text{ если } m < n ,$$

$$S = [S^{'};0]^T, \text{ если } m > n ,$$

$$S = S^{'}, \text{ если } m = n ,$$

$$S^{'} = diag\{s_1, s_2, ..., s_r,\ s_1 \geq s_2 \geq ... \geq s_r\}.$$

Числа s_i, $i = 1, 2, ..., r$ называются сингулярными числами матрицы A, которые определяются матрицей A однозначно.

Сингулярное разложение вещественной прямоугольной матрицы A в покомпонентной форме имеет следующее представление:

$$A = s_1 U_1 V_1^T + s_2 U_2 V_2^T + ... + s_r U_r V_r^T, \qquad (4.10)$$

где s_i –сингулярные числа матрицы A, U_i, V_i – соответственно, правые и левые сингулярные векторы, r –ранг матрицы. Эти сингулярные числа и сингулярные векторы удовлетворяют следующим соотношениям:

$$s_1 \geq s_2 \geq ... \geq s_r \geq 0,\ s_i = U_i^T A V_i,\ U_i^T U_i = 1,\ V_i^T V_i = 1, i = 1,..., r. \qquad (4.11)$$

Известно, что процессы сингулярного разложения для любой вещественной матрицы A обладают весьма полезными свойствами для теории и приложений, а именно, каждая матрица над полем вещественных чисел имеет вещественные сингулярные числа и векторы. Кроме того, сингулярное разложение матриц устойчиво к малым возмущениям матриц, т.е. сингулярное разложение каждой матрицы является хорошо обусловленной процедурой.

Относительно практических аспектов, сингулярное разложение матрицы в общем случае может быть получено по достаточно простой и надежной схеме:

$$\begin{gathered} V_{(k+1)}^T = U_{(k)}^T A, \\ V_{(k+1)} = V_{(k+1)} / \left| V_{(k+1)} \right| \\ U_{(k+1)} = A V_{(k+1)},\ \ U_{(k+1)} = U_{(k+1)} / \left| U_{(k+1)} \right| \\ s_k = U_k^T A V_k,\ \left| s_{k+1} - s_k \right| \leq \varepsilon, \end{gathered} \qquad (4.12)$$

где $k=0,1,2,...$ – номер итерации, $\left| U_{(k+1)} \right|$ - любая векторная норма, ε - заданная точность вычисления. Можно показать, что для произвольных начальных векторов $U_{(0)}$, $V_{(0)}$ итерации по схеме (4.12) сходятся в общем случае к сингулярным векторам U, V, соответствующим максимальному сингулярному числу $s_{\max} = U^T A V$.

Следуют отметить, что такие свойства не свойственны спектральному разложению, которое в действительности формирует основу для многомерного статистического анализа. В отличие от сингулярного разложения матриц, собственные числа и собственные векторы спектрального разложения являются вещественными только

для вещественных симметрических матриц, в общем случае не симметрические вещественные матрицы обладают комплексным спектром и определить его не просто.

С использованием вышеприведенного итеративного алгоритма (4.12) сингулярное разложение матрицы A, представленное в форме (4.10), (4.11) может быть получено с использованием метода исчерпывания.

Сущность этого метода заключается в следующем:

- максимальное сингулярное число и соответствующие ему правый и левый сингулярный векторы матрицы A вычисляются с помощью итеративного алгоритма (4.12). Формируется матричная компонента

$$A_1 = s_1 U_1 V_1^T ;$$

- формируется матрица невязки

$$A_2 = A - A_1 = A - s_1 U_1 V_1^T \qquad (4.13)$$

для которой максимальное сингулярное число и соответствующие ему правый и левый сингулярный векторы матрицы A_2 вычисляются с помощью итеративного алгоритма (4.12) и т.д.

Данный алгоритм иммунокомпьютинга может быть рассмотрен как «иммунный» алгоритм, так как любой образ может быть представлен как частный случай формального протеина и его распознавание основывается на энергии связи с антителом формального протеина.

Таким образом, иммунокомпьютинг позволяет с большой эффективностью решать следующие задачи: обучения с экспертом, самообучения (обучения без эксперта), группировки и классификации, представления результатов вычислений в пространстве образов.

4.5.3. Применение искусственных иммунных систем к решению задачи классификации

В ИИС используется способность естественной иммунной системы вырабатывать новые типы антител и отбирать наиболее подходящие из них для взаимодействия с попавшими в организм антигенами [184]. Для объяснения иммунологических механизмов существуют различные теории – теория иммунной сети, принципы клонального и негативного отбора.

Защитная реакция организма в борьбе с болезнью состоит в том, что он начинает вырабатывать клетки (антитела), способные распознать и нейтрализовать антигены. Клетки, полученные в результате мутационного процесса, имеют большое сходство с антигенами, имеют большее время жизни и остаются в организме на случай, если в будущем атака повторится (память клетки). Пока процент клонирования прямо пропорционален сходству с антигенами, процент мутации обратно пропорционален такой схожести, поэтому, чем ближе клетка к антигену, тем меньше процент ее мутации. С другой стороны, если сходство антигена и клетки очень низкое, высокий процент мутации применяется в надежде повысить значение сходства [185-188].

Для эволюционных алгоритмов известно [189], что сходимость к глобальному оптимуму в задаче оптимизации достигается в том случае, если есть уверенность в том, что алгоритм находит решение за конечное число шагов, и если такое решение будет оставаться в дальнейшем в популяции. Поскольку состояния переходов эволюционных алгоритмов имеют стохастический характер, детерминированная концепция сходимости не может быть использована для определения срока действия таких алгоритмов. Существуют две широко используемые меры стохастической сходимости эволюционных алгоритмов – это полное совпадение и совпадение по значению [189]:

Определим задачу классификации как отображение $f(X) \to \{1,...,c\}$ любого образа *X* в одно из целых чисел *1,…,c*, которые представляют классы.

Задача классификации может быть сформулирована следующим способом.

Дано:

- число классов *c*;
- набор из *m* обучающих образов: $X_1, X_2,..., X_m$;
- класс любого обучающего образа: $f(X_1) = c_1,..., f(X_m) = c_m$;
- произвольный *n*-мерный вектор *Z*.

Найти:

Класс вектора *Z: f(Z)=?*

Процесс обучения состоит из следующих этапов:

1. Сформировать обучающую матрицу $A = [X_1,..., X_m]^T$ размерности $m \times n$.

2. Вычислить максимальное сингулярное число *s*, а также левый и правый сингулярные векторы *L* и *R* обучающей матрицы по следующей итеративной (эволюционной) схеме:

$$L_{(0)} = [1,...,1]^T,\ R^T = L_{(k-1)}^T A,\quad R_{(k)} = R/|R|,\ \text{где}\ |R| = \sqrt{r_1^2 + ... + r_n^2},$$

$$L = AR_{(k)},\quad L_{(k)} = L/|L|,$$

где $|L| = \sqrt{l_1^2 + ... + l_n^2}$, $s_{(k)} = L_{(k)}^T A R_{(k)}$, $k = 1,2,...,$

до выполнения условия $|s_{(k)} - s_{(k-1)}| < \varepsilon$, $s = s_{(k)}$, $L = L_{(k)}$, $R = R_{(k)}$.

3. Хранить сингулярное число s.

4. Хранить правый сингулярный вектор *R* (как «антитело-пробу»).

5. Для всякого *i=1,...,m* хранить компоненту l_i левого сингулярного вектора *L* и класс c_i соответствующий обучающему образу x_i.

6. Для всякого *n*-мерного образа *Z* вычислить его энергию связи с *R*:

$$w(z) = Z^T R / s.$$

7. Выбрать l_i, которая имеет минимальное расстояние *d* с *w*:

$$d = \min_i |w - l_i|,\ i=1,...,m,$$

и считать класс c_i искомым классом образа *Z*.

В общем виде один шаг работы иммунного алгоритма можно представить следующим образом [184]:

$$\forall i \in \{1,...,m) : x_i^{'} = mut(clon(x_1,...,x_n)),\ (x_1^{''},...,x_k^{''}) = aging(x_1,...,x_n,x_1^{'},...,x_m^{'}),$$

$$(y_1,...,y_n) = sel(x_1^{''},...,x_k^{''}),$$

где $(x_,...,x_n) \in X^n$ – текущая популяция антител; $(x_1^{'},...,x_m^{'})$ – популяция антител, возникающая в результате клонирования и мутации; $(x_1^{''},...,x_k^{''})$ – антитела, которые удаляются из популяции; $(y_1,...,y_n)$ – антитела, добавляемые в текущую популяцию.

Как следует из [185], ИА присущи следующие операторы: клонирование, мутация, старение и селекция. Рассмотрим подробнее эти операторы.

Оператор клонирования генерирует новое поколение копий антител в будущей популяции. Известны следующие основные операторы клонирования [185]: а) статический оператор клонирования, который просто копирует каждую В-клетку, производя переходную популяцию; б) пропорциональный оператор клонирования, который клонирует В-клетки пропорционально их антигенной схожести; в) оператор вероятностного клонирования, в

соответствии с которым В-клетки выбираются из текущего поколения в зависимости от вероятности клональной селекции.

Оператор мутации действует в зависимости от имеющейся популяции клонов, применяя к каждому антителу определенное количество одиночных мутаций, осуществляемых случайным образом. Можно выделить следующие способы мутации [191]:

1. Статическая мутация. Количество мутаций зависит от минимизируемой функции f , поэтому антитело в каждый момент времени будет подвергаться определенному числу мутаций.

2. Пропорциональная мутация. Количество мутаций антитела пропорционально соответствующему значению.

3. Обратно пропорциональная мутация. Количество мутаций антитела обратно пропорционально соответствующему значению.

4. Круглая мутация. Каждое антитело подвергается мутации круглого сочетания.

Оператор старения устраняет старые особи. Статический оператор старения использует возрастной параметр для максимального количества поколений антител, которым разрешено оставаться в популяции. Когда антитело старше, оно удаляется из системы, даже если оно может оказаться вполне пригодным на последующих итерациях.

При клональной экспансии клонированное антитело наследует Элитный вариант этого оператора получается взятием лучшего антитела из популяции в поколение.

Оператор селекции заменяет наихудшие антитела в популяции новыми случайными антителами.

В работе рассмотрены обобщенные условия сходимости ИА в ависимости от используемых иммунных операторов. Анализ сходимости ИА основан на выполнении двух условий:

1) в результате мутации можно достичь оптимальное состояние из неоптимального за один шаг;

2) как только оптимальное состояние будет найдено, оно сохранится в популяции и не будет утеряно. Показано, что только операторы мутации и селекции, которые могут вносить изменения в антитела, способствует поиску оптимума.

4.6. Применение природных вычислений к решению задач оптимизации

Как известно, задача маршрутизации является NP-трудной задачей, что не позволяет найти оптимальное решение за короткое (приемлемое) время для задач средней и большой размерности. Поэтому вместо точных, как правило, используются приближенные алгоритмы, которые позволяют найти близкое к оптимальному рациональное решение.

Для нахождения приближенных решений применяется множество алгоритмов, четыре из которых рассматриваются в данной работе: искусственные нейронные сети, генетический алгоритм, муравьиный алгоритм и новый подход, основанный на искусственных иммунных системах (ИИС). Все эти методы относятся к направлению «природных вычислений», т.е. моделируют те или иные биологические процессы, алгоритмы которых природа создавала миллионы лет. Стоит отметить, что эффективность того или иного алгоритма зависит от характеристик исходных данных задачи, поэтому нельзя однозначно определить, какой из алгоритмов наиболее эффективен.

Математическая модель задачи имеет следующий вид:

$$R = \sum_{i=1}^{n}\sum_{j=1}^{n} r_{ij} z_{ij} \to \max. \tag{4.14}$$

$$\begin{cases} \sum_{j=1}^{n} z_{ij} = 1,\ i = \overline{1,n}; \\ \sum_{i=1}^{n} z_{ij} = 1,\ j = \overline{1,n}; \\ z_{ij} \in \{0;1\},\ i = \overline{1,n} \quad j = \overline{1,n}. \end{cases} \tag{4.15}$$

4.6.1. Нейронные сети

Хопфилд и Танк показали подход к ее приближенному решению на основе сетей Хопфилда [42,124]. Рассмотрим вкратце этот подход. Для описания возможных маршрутов авторы предложили специальный тип матрицы. В ней города образуют строки, а столбцы отображают последовательность городов в маршруте. В позиции (*x*, *i*) матрицы стоит 1 в том случае, когда город *x* занимает *i*-е место в маршруте.

В случае *n* городов существует $\frac{n1}{2n}$ различных маршрутов, среди которых необходимо найти кратчайший. Для получения решения задача отображается сетью Хопфилда.

В ней каждый нейрон обозначается двумя индексами *x* и *i*, причем *x* отражает город, а *i*-ю позицию в маршруте, т.е. z_{xi} – это выход нейрона, в котором город *x* размещен на *i*-й позиции маршрута.

К энергетической функции *E* сети Хопфилда предъявляются следующие условия [124]:

- должна быть минимальна только для допустимых решений, которые содержат одну единицу в каждой строке и в каждом столбце матрицы описания маршрутов;
- для решений с более короткими маршрутами должна принимать меньшие значения.

Энергетическая функция, удовлетворяющая этим условиям, может иметь вид [124]:

$$E = \frac{A}{2}\sum_x \sum_i \sum_{j\neq i} z_{xi}z_{xj} + \frac{b}{2}\sum_x \sum_i \sum_{y\neq x} z_{yi}z_{xi} + \frac{C}{2}\left(\left(\sum_x \sum_i z_{xi}\right) - n\right)^2$$
$$+ \frac{D}{2}\sum_x \sum_i \sum_y r_{xy}z_{xi}(z_{yi+1} + z_{yi-1})$$

При ее выборе учтены следующие соображения:

- первое слагаемое равно нулю только в тех случаях, когда каждая строка матрицы описания маршрутов содержит лишь одну единицу;
- второе слагаемое равно нулю только тогда, когда каждый столбец матрицы содержит лишь одну единицу;
- третье слагаемое равно нулю лишь в тех случаях, когда в матрице описания маршрутов имеется *n* единиц, что означает: каждый город посещается лишь один раз;

• четвертое слагаемое отражает длину маршрута. Обратим внимание, что для каждого города *x*, расположенного на *i*-й позиции, определяется расстояние *rxy* до его последователя *y* на позиции *i*+1 и его предшественника *y* на позиции *i*–1.

Применение сетей Хопфилда ограничивается высокой вычислительной сложностью (n^4, где n – размерность задачи) и необходимостью тщательного подбора параметров функции энергии и функции активации нейрона [124]. Ситуацию можно существенно упростить, если применить следующий подход.

Для решения задачи (4.14)-(4.15) предложена рекуррентная нейронная сеть [42,82,87, 125, 157], которая описывается дифференциальным уравнением

$$\frac{\partial u_{ij}(t)}{\partial t} = -\eta\left(\sum_{k=1}^{n} z_{ik}(t) + \sum_{l=1}^{n} z_{lj}(t) - 2\right) + \lambda r_{ij} \exp\left(-\frac{t}{\tau}\right), \qquad (4.16)$$

где

$$z_{ij} = f(u_{ij}(t)), \quad f(u) = \frac{1}{1+\exp(-\beta u)}.$$

Как и в сети Хопфильда, здесь используется матрица нейронов размером $n \times n$, но нейроны взаимодействуют не по принципу «каждый с каждым», а по строкам и столбцам.

Разностный вариант этого уравнения имеет вид

$$u_{ij}^{t+1} = u_{ij}^{t} - \Delta t \cdot \left[\eta\left(\sum_{k=1}^{n} z_{ik}(t) + \sum_{l=1}^{n} z_{lj}(t) - 2\right) - \lambda r_{ij} \exp\left(-\frac{t}{\tau}\right)\right], \qquad (4.17)$$

где Δt - шаг по времени. Параметры $\Delta t, \eta, \lambda, \tau, \beta$ подбираются экспериментально и существенно влияют на скорость достижения решения задачи и качество этого решения.

Для ускорения решения системы уравнений (4.17) предложен принцип «Winner takes all» [42, 125,157]:

1. Порождается матрица $\|z_{ij}^0\|$ случайных значений $z_{ij}^0 \in [0,1]$. Итерация (4.17) продолжаются до тех пор, пока не выполнится неравенство

$$\sum_{k=1}^{n} z_{ik}(t) + \sum_{l=1}^{n} z_{lj}(t) - 2 \le \varepsilon,$$

где ε - заданная точность выполнения ограничений (4.15).

2. Выполняется преобразование поученной матрицы решения $\|z_{ij}\|$:

2.1. $i = 1$.

2.2. В i-й строке матрицы отыскивается максимальный элемент $z_{i,j_{\max}}$, $j_{\max}$ - номер столбца с максимальным элементом.

2.3. Выполняется преобразование $z_{i,j_{\max}}=1$. Все остальные элементы первой строки и столбца обращаются в нуль:

$$z_{ij}=0,\ j\neq j_{\max},$$

$$z_{k,j_{\max}}=0,\ k\neq i.$$

Далее происходит переход к строке $j_{\max}$.

Действия 2.2. и 2.3 повторяются, пока не произойдет возврат к первой строке, что будет означать завершение построения цикла

3. Если возврат к строке 1 произошел раньше, чем в матрице $\|z_{ij}\|$ значение 1 получили n элементов, то это означает, что длина построенного цикла меньше n. В этом случае шаги 1 и 2 повторяются.

При таком подходе вычислительная сложность алгоритма решения задач маршрутизации снижается с $O(n^4)$ до $O(n^2)$.

4.6.2. Генетический алгоритм

Генетические алгоритмы являются одним из видов эволюционных вычислений. Как правило, они применяются для поиска решений в задачах с большой комбинаторной сложностью, для которых трудно найти решение аналитическим путем. Принцип работы генетического алгоритма основан на теории эволюции Чарльза Дарвина. Основателем генетических алгоритмов считается Джон Холланд (John Holland), опубликовавший в 1975 году первую книгу в этой области исследований под названием “Адаптация в естественных и искусственных системах” (“Adaptation in Natural and Articial Systems”) [82,87].

В классическом генетическом алгоритме решение кодируется двоичной последовательностью. Такой способ подходит для случаев, когда изменение одного бита в решении в большинстве случаев приводит к незначительному изменению целевой функции.

В задачах на графах решением является маршрут - последовательность вершин. Изменение одной вершины в маршруте может кардинально отразиться на значении целевой функции. Поэтому в подобных задачах операции производятся над последовательностями целых чисел, обозначающих номера вершин. Функцией приспособленности является длина маршрута [42].

Основная идея генетического алгоритма состоит в преобразовании задачу (4.14) – (4.15) в последовательность задач безусловной минимизации

$$f_k(z) = f(x) + \frac{1}{k}\left(\sum_{i=1}^{n} z_{ij} + \sum_{j=1}^{m} z_{ij} - 2 \right) \to \min, \qquad k = 1,2,\ldots .$$

Переформулируем задачу оптимизации как задачу нахождения минимума некоторой функции $f_k(z_1, z_2, \ldots, z_n)$, называемой *функцией приспособленности* (fitness function). Она должна принимать неотрицательные значения на ограниченной области определения (для того, чтобы мы могли для каждой особи считать её приспособленность, которая не может быть отрицательной), при этом совершенно не требуются непрерывность и дифференцируемость.

Шаг алгоритма состоит из трех стадий [82,87]:

1. Генерация промежуточной популяции (*intermediate generation*) путем отбора (*selection*) текущего поколения.

2. Скрещивание (*recombination*) особей промежуточной популяции путем *кроссовера* (*crossover*), что приводит к формированию нового поколения.

Кроссинговер заключается во взаимном обмене генами между «родителями» - хромосомами предварительно выбранной пары.

Предварительно задается величина P_K – вероятность кроссинговера и вводится флажок FG с двумя состояниями «выполнять», «не выполнять». Исходное состояние FG «не выполнять». При выполнении кроссинговера последовательно просматриваются локусы выбранной пары хромосом. С вероятностью P_k «флажок» FG переходит в состояние «выполнять». Если FG перешел в состояние «выполнять», то производится обмен генами между парой хромосом в текущем локусе, далее «флажок» переходит в состояние «не выполнять», а затем осуществляется переход к следующему локусу.

3.Мутация нового поколения.

Операция мутации заключается в изменении значения гена. Алгоритм мутации реализуется следующим образом.

Последовательно выбираются хромосомы из текущей популяции. В пределах выбранной хромосомы последовательно просматриваются гены. После перехода к очередному гену, FG с вероятностью P_M переходит в состояние «выполнять». Если FG перешел в состояние «выполнять», то случайным образом ген g_n

принимает одно из значений в заданном диапазоне, за исключением значения, которое ген имеет перед мутацией. Далее FG переходит в состояние «не выполнять» и выбирается следующий ген хромосомы, или следующая хромосома.

4.6.3. Муравьиный алгоритм

Муравьиные алгоритмы основаны на моделировании взаимодействия муравьев в муравьиной колонии. Муравьиные алгоритмы эффективны при решении различных комбинаторных задач на графах, включая задачу о коммивояжере. Их исследование началось в начале 90-х годов Марко Дориго в Университете Брюсселя.

Муравьиная колония представляет собой сложную распределенную не централизованную систему. Каждый муравей (по сути является агентом в мультиагентной системе) выполняет простые однообразные действия, обладая минимумом информации и взаимодействую с небольшим количеством других муравьев и небольшими участками окружающей среды. Однако вся система в целом решает сложные задачи оптимизации маршрутов, находя очень близкие к оптимальным решения, адаптируясь к внешним изменяющимся условиям.

Взаимодействие между муравьями происходит по средствам прямого и непрямого обмена информацией. Прямой обмен заключается в непосредственном контакте, а непрямой в изменении окружающей среды одним муравьем и распознавание этих изменений другими. На основе непрямого обмена и моделируются муравьиные алгоритмы. Для изменения окружающей среды муравьи распыляют феромон, запах которого остается в почве некоторое время.

На начальном этапе работы алгоритма муравьи выбирают направления случайным образом, пытаясь достичь цели (найти пищу), помечая феромонами свой путь. Те муравьи, которые раньше всех достигнут цели, соответственно вернуться обратно раньше всех тем же путем, увеличив содержание феромонов на этом маршруте. Через несколько итераций кратчайший маршрут будет сильно отличаться от всех остальных вариантов пути. Адаптивность к внешним изменениям осуществляется за счет испарения феромонов. Если на кратчайшем пути встречается препятствие, муравей случайным образом выбирает другой путь, таким образом через некоторое время находится новый кратчайший маршрут.

Муравьиный алгоритм, в отличие от генетического, гораздо быстрее адаптируется к изменению внешних условий.

4.6.4. Реализация алгоритма искусственных иммунных систем для решения задач маршрутизации

Для решения задачи маршрутизации с помощью алгоритмов искусственной иммунной системы (ИИС) необходимо сопоставить биологические объекты и процессы их математическим аналогам.

В настоящее время активно развиваются искусственные иммунные системы (ИИС) как новое направление в области вычислительного интеллекта. При этом иммунные алгоритмы (ИА) широко используются в различных областях интеллектуальной обработки информации. Свойства ИИС, такие как распознавание, разнообразие, обучение, память, распределенное обнаружение и др., позволяют использовать иммунные принципы для решения таких задач как распознавание образов, поиск данных, компьютерная безопасность, обнаружение ошибок, классификация, оптимизация и др. [128].

В ИИС используется способность естественной иммунной системы вырабатывать новые типы антител и отбирать наиболее подходящие из них для взаимодействия с попавшими в организм антигенами. Для объяснения иммунологических механизмов существуют различные теории – теория иммунной сети, принципы клонального и негативного отбора.

Естественная иммунная система состоит из большого количества защитных элементов, молекул и органических соединений, поддерживающих организм в здоровом состоянии, борясь с болезнями, служащими причиной заболеваний. Защитные элементы, используемые в естественной иммунной системе, называются лимфоцитами (белые кровяные тельца), главная задача которых – борьба с антигенами, молекулами, принадлежащими чужеродным телам, таким, как бактерии или вирусы, которые внедрились в организм.

Защитная реакция организма в борьбе с болезнью состоит в том, что он начинает вырабатывать клетки (антитела), способные распознать и нейтрализовать антигены. Клетки, полученные в результате мутационного процесса, имеют большое сходство с антигенами, имеют большее время жизни и остаются в организме на случай, если в будущем атака повторится (память клетки). Пока

процент клонирования прямо пропорционален сходству с антигенами, процент мутации обратно пропорционален такой схожести, поэтому, чем ближе клетка к антигену, тем меньше процент ее мутации. С другой стороны, если сходство антигена и клетки очень низкое, высокий процент мутации применяется в надежде повысить значение сходства.

Для эволюционных алгоритмов известно [128], что сходимость к глобальному оптимуму в задаче оптимизации достигается в том случае, если есть уверенность в том, что алгоритм находит решение за конечное число шагов, и если такое решение будет оставаться в дальнейшем в популяции. Поскольку состояния переходов эволюционных алгоритмов имеют стохастический характер, детерминированная концепция сходимости не может быть использована для определения срока действия таких алгоритмов. Существуют две широко используемые меры стохастической сходимости эволюционных алгоритмов – это полное совпадение и совпадение по значению [128]:

В общем виде один шаг работы иммунного алгоритма можно представить следующим образом :

$$\forall i \in \{1,...,m\} : x_i^{'} = mut(clon(x_1,...,x_n)),$$

$$(x_1^{''},...,x_k^{''}) = aging(x_1,...,x_n,x_1^{'},...,x_m^{'}),$$

$$(y_1,...,y_n) = sel(x_1^{''},...,x_k^{''}),$$

где $(x_,...,x_n) \in X^n$ – текущая популяция антител; $(x_1^{'},...,x_m^{'})$ – популяция антител, возникающая в результате клонирования и мутации; $(x_1^{''},...,x_k^{''})$ – антитела, которые удаляются из популяции; $(y_1,...,y_n)$ – антитела, добавляемые в текущую популяцию.

ИА присущи следующие операторы: клонирование, мутация, старение и селекция. Рассмотрим подробнее эти операторы.

Оператор клонирования генерирует новое поколение копий антител в будущей популяции. Известны следующие основные операторы клонирования: а) статический оператор клонирования, который просто копирует каждую В-клетку, производя переходную популяцию; б) пропорциональный оператор клонирования, который клонирует В-клетки пропорционально их антигенной схожести; в) оператор вероятностного клонирования, в соответствии с которым В-клетки выбираются из текущего поколения в зависимости от вероятности клональной селекции.

Оператор мутации действует в зависимости от имеющейся популяции клонов, применяя к каждому антителу определенное количество одиночных мутаций, осуществляемых случайным образом.

Оператор старения устраняет старые особи. Статический оператор старения использует возрастной параметр для максимального количества поколений антител, которым разрешено оставаться в популяции. Когда антитело старше, оно удаляется из системы, даже если оно может оказаться вполне пригодным на последующих итерациях.

При клональной экспансии клонированное антитело наследует возраст его родителя. После этапа мутации только те антитела, которые получили высшее значение аффинности, получат возраст, равный 0. Элитный вариант этого оператора получается путем взятия наилучших антител популяции в поколение с возрастом, равным 0.

Элитный вариант этого оператора получается взятием лучшего антитела из популяции в поколение.

Оператор селекции заменяет наихудшие антитела в популяции новыми случайными антителами.

Для решения задачи маршрутизации с помощью алгоритмов искусственной иммунной системы необходимо сопоставить биологические объекты и процессы их математическим аналогам.

Антигены, обозначающие в терминах иммунной системы вещества из внешней среды, соответствуют условиям задачи – набору вершин графа. В-лимфоциты соответствуют агентам, перемещающимся по вершинам графа, клонирующие и уничтожающие себя, использую алгоритмы положительного и отрицательного отбора. Агент начинает свой путь в начальной вершине, при каждой итерации алгоритма он имеет возможность клонировать себя. Попадая в вершину, которую он уже посещал, агент уничтожает себя согласно правила положительного отбора. После завершение обхода по правилу отрицательного отбора выбирается агент с наименьшей длиной пути.

Очевидно, что популяция агентов будет расти экспоненциально. Чтобы этого избежать, введем параметр, ограничивающий максимальное количество агентов: клонирование новых агентов будет происходить только в том случае, если в популяции есть свободные места. Для сокращения количества бесполезных агентов, длина маршрутов которых превысили текущий наилучший результат

до завершения пути, будем записывать текущую минимальную длину пути при создании нового агента. Если путь агента превышает это значение, он самоуничтожается.

Как правило, оптимальный путь в задаче маршрутизации состоит из ребер, соединяющих ближайшие вершины, поэтому логичнее будет выбирать новые вершины не с равной вероятностью, а с зависящей от удаления от текущей (чем дальше, тем меньше вероятность). Агент, запущенный для прохождения пути повторно, содержит свой предыдущий маршрут.

Алгоритм действий одного агента в псевдокоде выглядит следующим образом:

1. Если маршрут уже проходился ранее - перейти в очередную вершину, в противном случае перейти в одну из вершин графа (вероятность выбора зависит от удаленности вершины от текущей);
2. Если агент уже был в этой вершине – самоуничтожиться;
3. Увеличить длину на размер пройденного пути;
4. Если длина превышает известный агенту минимум – самоуничтожится;
5. Если в популяции есть свободные места - клонировать себя, изменив последнюю вершину у клона на случайную;
6. Перейти к шагу 1.

Рассмотренный в данной работе алгоритм и программа, основанный на искусственных иммунных системах, применительно к классической задачи маршрутизации показали хорошие результаты (рис.4.8). Интерфейс программы включает характеристики иммунного алгоритма в соответствии с вариантом, сведения о разработчике, краткую справку (руководство пользователя).

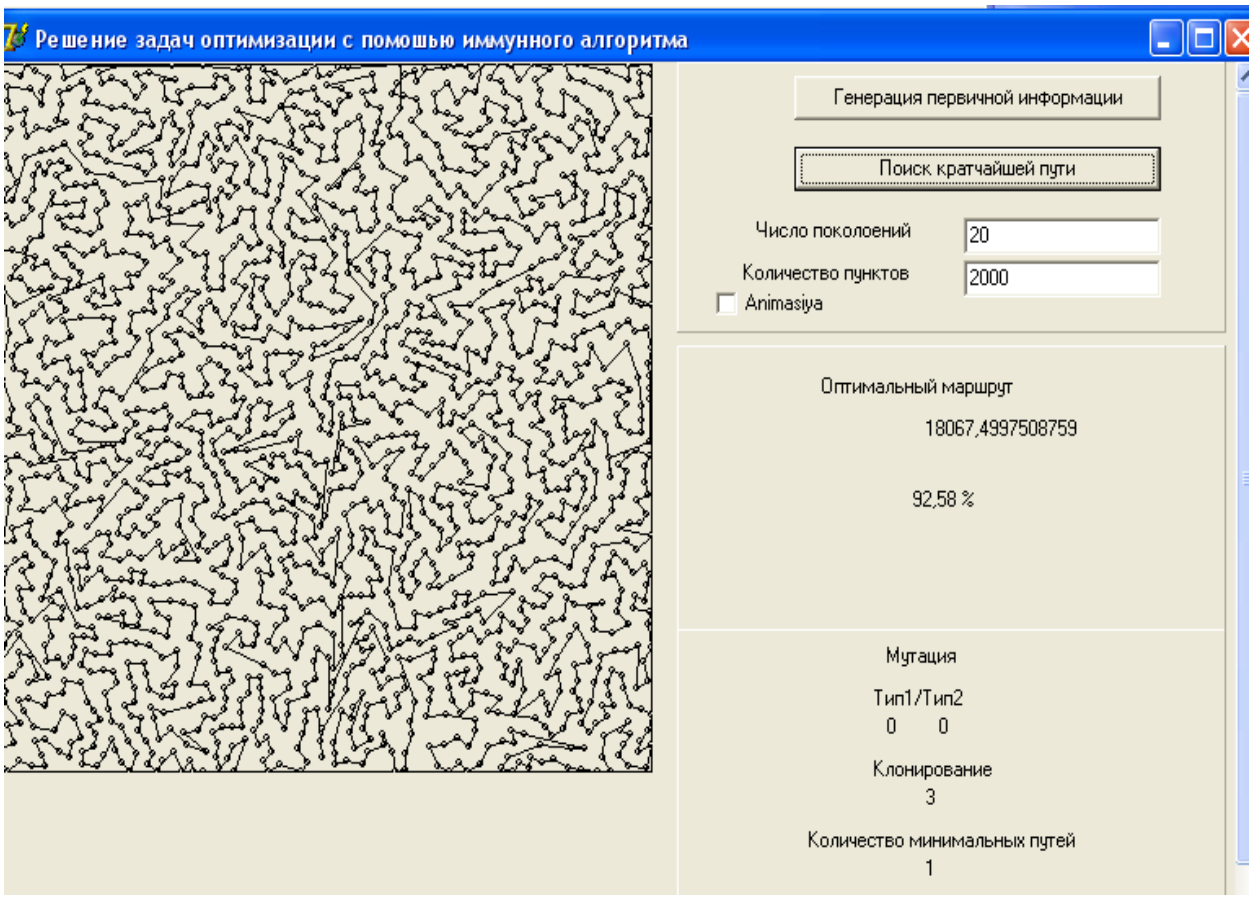


Рис.4.8. Решение задач маршрутизации с помощью иммунного алгоритма

Анализируя полученные результаты моделирования приходим к выводу, что оптимальным количеством маршрутов можно считать 200, число поколений, нет необходимости повторять алгоритм больше 500 раз (поколений), чтобы получить хороший результат.

В дальнейшем планируется продолжить исследование применения этих методов к задачам маршрутизации, их динамическим и распределенным вариантам.

4.6.5. Генетический алгоритм решения задачи оптимального использования торговых агентов

Торговая фирма продает товары в n различных городах, покупательная способность жителей которых оценивается в b_j условная единица, $j=\overline{1,n}$. Для реализации товаров фирма располагает n торговыми агентами, каждого из которых она направляет в один из городов. Профессиональный уровень агентов различен; доля реализуемых i-м торговым агентом покупательных способностей составляет a_i, $i=\overline{1,m}$. Как следует распределить торговых агентов по городам, чтобы фирма получила максимальную выручку от продажи товаров?

Оптимальное решение этой проблемы может быть найдено с помощью задачи о назначениях. В качестве кандидатов выступают торговые агенты, в качестве работ — города.

Введем параметр $c_{ij}=a_i b_j$, характеризующий величину покупательных способностей, реализуемых i-м торговым агентом в j-м городе.

Управляющие переменные x_{ij}, $i=\overline{1,n};\ j=\overline{1,n}$ определяются по формуле

$$x_{ij}=\begin{cases}1, \textit{ если } i-\textit{й агент направлен в } j-\textit{й город},\\ 0, \textit{ в противном случае.}\end{cases}$$

Математическая модель запишется в следующей форме:

$$C=-\sum_{i=1}^{n}\sum_{j=1}^{n}c_{ij}x_{ij}\to min. \tag{4.18}$$

$$\begin{cases}\sum\limits_{j=1}^{n}x_{ij}=1,\ i=\overline{1,n};\\ \sum\limits_{i=1}^{n}x_{ij}=1,\ j=\overline{1,n};\\ x_{ij}\in\{0;1\},\ i=\overline{1,n}\quad j=\overline{1,n}.\end{cases} \tag{4.19}$$

Первое и второе ограничения формализуют соответственно условия о том, что в каждый город направляется один торговый агент, и один торговый агент не может работать в двух городах. Целевая функция С— это сумма реализованных покупательных способностей

всеми торговыми агентами во всех городах. Она должна быть максимальна.

Решить задачу о назначениях – значит найти x_{ij} , удовлетворяющие (4.19) и доставляющие минимум функции (4.18).

Благодаря своему широкому применению, теория о нахождении кратчайших путей в последнее время интенсивно развивается.

Нахождение кратчайшего пути – жизненно необходимо и используется практически везде, начиная от нахождения оптимального маршрута между двумя объектами на местности (например, кратчайший путь от дома до университета), в системах автопилота, для нахождения оптимального маршрута при перевозках, коммутации информационного пакета в Internet и т.п.

Кратчайший путь рассматривается при помощи некоторого математического объекта, называемого графом.

Существуют три наиболее эффективных алгоритма нахождения кратчайшего пути:

1. алгоритм Дейкстры (используется для нахождения оптимального маршрута между двумя вершинами);
2. алгоритм Флойда (для нахождения оптимального маршрута между всеми парами вершин);
3. алгоритм Йена (для нахождения k-оптимальных маршрутов между двумя вершинами).

Указанные алгоритмы легко выполняются при малом количестве вершин в графе. При увеличении их количества задача поиска кратчайшего пути усложняется. Здесь на помощь приходит генетический алгоритм.

При разработке генетических процедур основное влияние уделялось разработке с учетом знаний о предметной области методов кодирования решений, модификации генетических операторов и организации эволюционного процесса.

Основная идея генетического алгоритма состоит в преобразовании задачу (4.18)–(4.19) в последовательность задач безусловной минимизации

$$f_k(x) = f(x) + \frac{1}{k}\left(\sum_{i=1}^{n} x_{ij} + \sum_{j=1}^{m} x_{ij} - 2\right) \to \min, \qquad k = 1,2,\ldots. \quad (4.20)$$

Переформулируем задачу оптимизации как задачу нахождения минимума некоторой функции $f(x_1, x_2, \ldots, x_n)$.

4.7. Оптимизация многоэкстремальных функций с помощью генетических алгоритмов

Рассматривается генетический алгоритм для решения многопараметрической непрерывной задачи оптимизации.

На практике подчас сложно, а порой и невозможно, зафиксировать свойства функциональной зависимости выходных параметров от входных величин, еще сложнее привести аналитическое описание такой зависимости.
В данной работе рассматривается один из таких генетических алгоритмов для решения многопараметрической непрерывной задачи оптимизации.

Оценка предлагаемого алгоритма проводилась на ряде тестовых функций [169]. Это прежде всего несколько тестовых функций De Jong'a, которые часто используются для проверки эффективности новых генетических алгоритмов, кроме того, были использованы сложные многоэкстремальные функции высокой размерности, практически нерешаемые классическими методами.

Все тестовые функции могут иметь различное число параметров (*d*). Поэтому имеет смысл запустить алгоритм для оптимизации некоторой функции сначала с небольшим d (например, 10 или 20), а затем с *d*=50, 100, 200,... Это даст возможность проверить масштабируемость алгоритма.

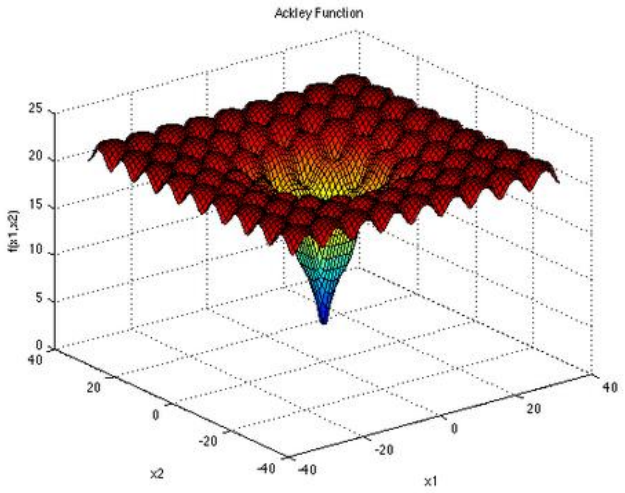
Эффективность работы ГА принято оценивать количеством вычислений целевой функции. Чем меньше, тем лучше. После некоторых функций приведены результаты работы ГА с искусственным отбором. Результаты целевой функции меньше 0.001 тоже засчитывались как найденный глобальный минимум.
Характеристики ГА:

- Фиксированный размер популяции;
- Фиксированная разрядность генов;
- Количество точек разрыва кроссовера равно числу генов (на каждый ген приходится ровно одна точка);
- Для скрещивания отбираются 50% популяции;
- Вероятность мутации 95%;
- Две "элитные" особи;
- Удаление одинаковых особей из популяции (с помощью мутации);
- Точность вычислений: 0.001;
- Результат подсчитан по 50 запускам алгоритма;

Т.к. генетические алгоритмы используют стохастичность, то для того, чтобы определить, насколько эффективен ГА нужно запустить его на одной и той же тестовой функции несколько раз и только после этого анализировать результат. Например, ГА для функции Растригина от 50 переменных находит глобальный минимум в ~40% случаев, используя не более 10000 вычислений функции.

Многоэкстремальные минимумы

1. Функция Ackley:



$$f(\mathbf{x}) = -a\exp\left(-b\sqrt{\frac{1}{d}\sum_{i=1}^{d}x_i^2}\right) - \exp\left(\frac{1}{d}\sum_{i=1}^{d}\cos(cx_i)\right) + a + \exp(1)$$

Рекомендуемые переменные: a = 20, b = 0.2 и c = 2π.

Область определения:

$x_i \in [-32.768, 32.768]$, для всех $i = 1, \ldots, d$

Глобальный минимум:

$$f(\mathbf{x}^*) = 0, \mathbf{x}^* = (0, \ldots, 0)$$

Метод Ньютона: xmin = 0.0000 0.0000; opt = 4.1940e-014;

Метод градиента xmin =0.0000 0.0000; opt = 1.9116e-011;

Метод наименьших квадратов и, как правило, дает наименьшее число итераций при минимизации.

xmin = 0.4120 0.3715; opt = 0.1446

Обратите внимание на то, что вопреки ожиданиям функция к успеху не привела. Выдано сообщение о превышении лимита числа итераций, а значения xmin явно далеки от верных.

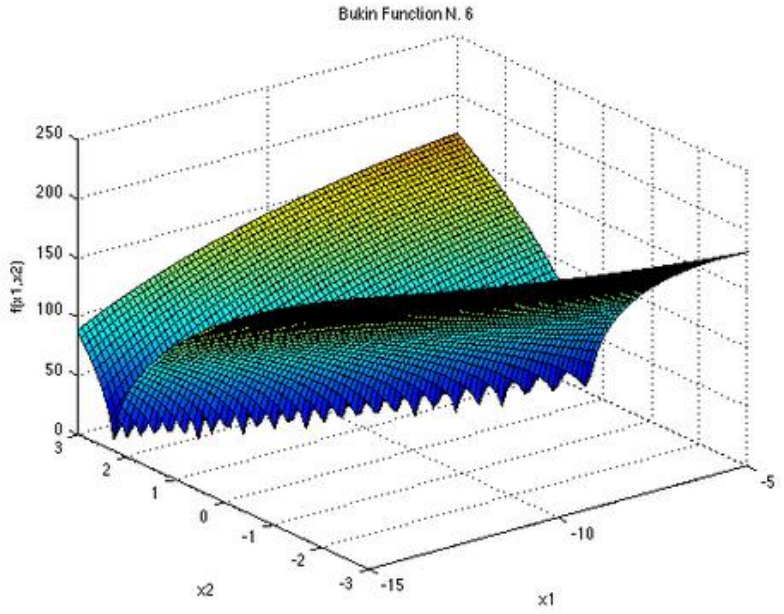
Генетический алгоритм:

d=10, количество нахождений глобального минимума 86%, число вычислений целевой функции не более 1250, максимальное значение 0,008325.

d=30, количество нахождений глобального минимума 84%, число вычислений целевой функции не более 1250, максимальное значение 0,108851.

d=50, количество нахождений глобального минимума 76%, число вычислений целевой функции не более 2500, максимальное значение 0,016291, для скрещивания отбиралось 40% популяции.

2. Функция Bukin:



$$f(\mathbf{x}) = 100\sqrt{\left|x_2 - 0.01x_1^2\right|} + 0.01|x_1 + 10|$$

Область определения:

$x_1 \in [-15, -5]$, $x_2 \in [-3, 3]$.

Глобальный минимум:

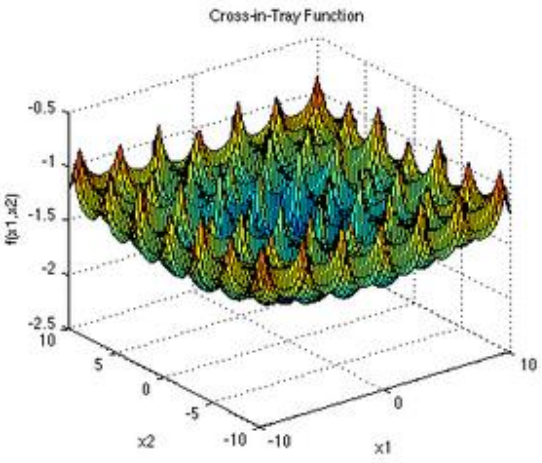
$$f(\mathbf{x}^*) = 0, \; \mathbf{x}^* = (-10, 1)$$

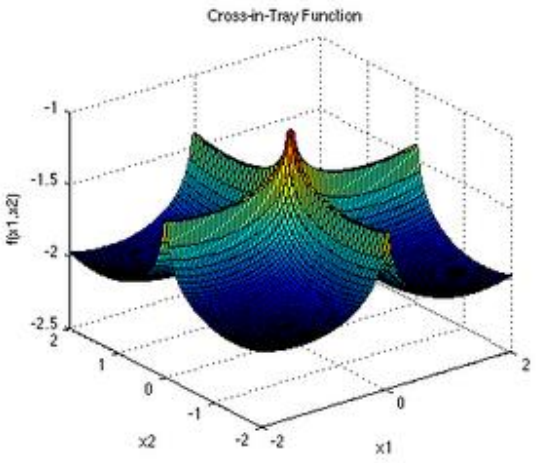
.

Генетический алгоритм:

d=2 количество нахождений глобального минимума 82%, число вычислений целевой функции не более 1250, максимальное значение 0,0025774.

3. Функция Cross-in-Tray:





$$f(\mathbf{x}) = -0.0001 \left(\left| \sin(x_1)\sin(x_2)\exp\left(\left| 100 - \frac{\sqrt{x_1^2 + x_2^2}}{\pi} \right| \right) \right| + 1 \right)^{0.1}$$

Область определения:

$x_i \in [-10, 10]$, для $i = 1, 2$.

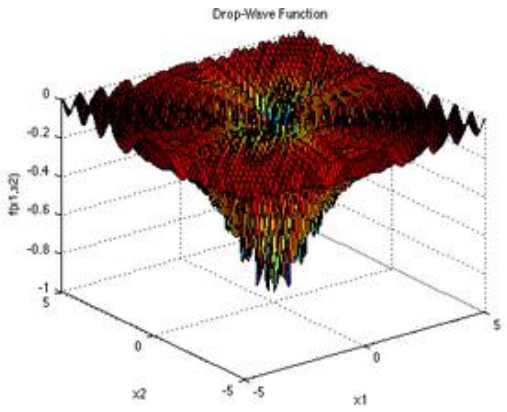
Глобальный минимум:

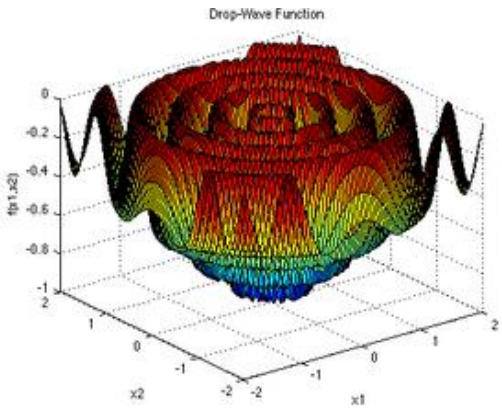
$f(\mathbf{x}^*) = -2.06261$, $\mathbf{x}^* = (1.3491, -1.3491), (1.3491, 1.3491), (-1.3491, 1.3491)$ и $(-1.3491, -1.3491)$

Генетический алгоритм:

d=2 количество нахождений глобального минимума 80%, число вычислений целевой функции не более 1250, максимальное значение -2,0627439.

4. Функция Drop-Wave:





$$f(\mathbf{x}) = -\frac{1 + \cos\left(12\sqrt{x_1^2 + x_2^2}\right)}{0.5(x_1^2 + x_2^2) + 2}$$

Область определения:

$x_i \in [-5.12, 5.12]$, для всех $i = 1, 2$.

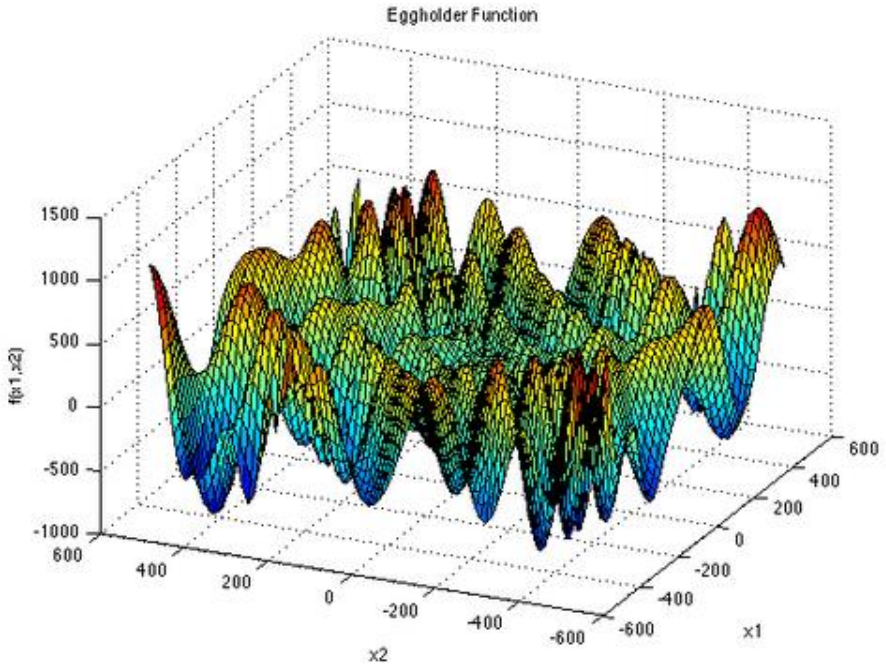
Глобальный минимум:

$$f(\mathbf{x}^*) = -1,\ \mathbf{x}^* = (0, 0)$$

Генетический алгоритм:

d=2 количество нахождений глобального минимума 80%, число вычислений целевой функции не более 1250, максимальное значение -1,003283.

5. Функция Eggholder:



$$f(\mathbf{x}) = -(x_2 + 47)\sin\left(\sqrt{\left|x_2 + \frac{x_1}{2} + 47\right|}\right) - x_1\sin\left(\sqrt{|x_1 - (x_2 + 47)|}\right)$$

Область определения:

$x_i \in [-512, 512]$, i = 1, 2.

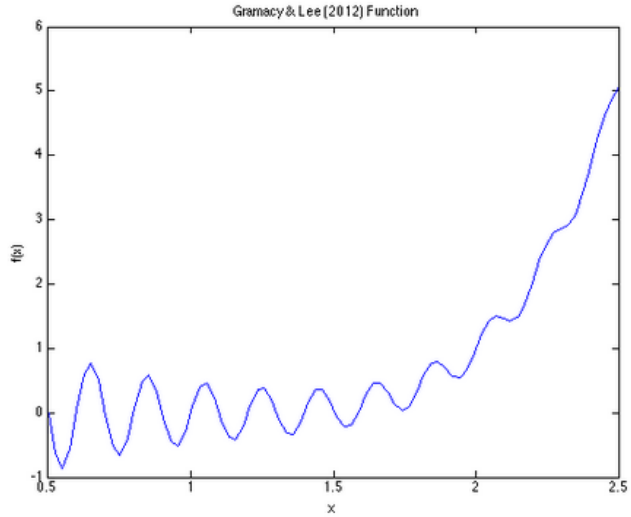
Глобальный минимум:

$$f(\mathbf{x}^*) = -959.6407,\ \mathbf{x}^* = (512, 404.2319)$$

Генетический алгоритм:

d=2 количество нахождений глобального минимума 80%, число вычислений целевой функции не более 1250, максимальное значение -959,644739.

6. Функция Gramacy & Lee (2012):

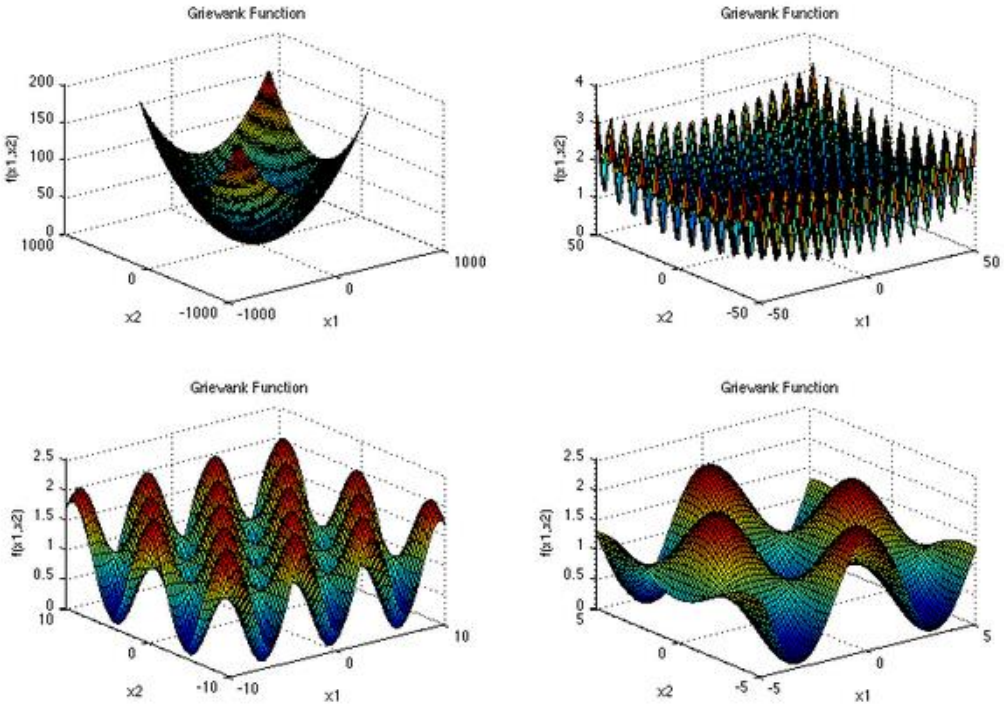


$$f(x) = \frac{\sin(10\pi x)}{2x} + (x-1)^4$$

Область определения:

$x \in [0.5, 2.5]$.

2. Функция Griewank:



$$f(\mathbf{x}) = \sum_{i=1}^{d} \frac{x_i^2}{4000} - \prod_{i=1}^{d} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$$

Область определения:

$x_i \in [-600, 600]$, i = 1, …, d.

Глобальный минимум:

$f(\mathbf{x}^*) = 0, \mathbf{x}^* = (0, \ldots, 0)$

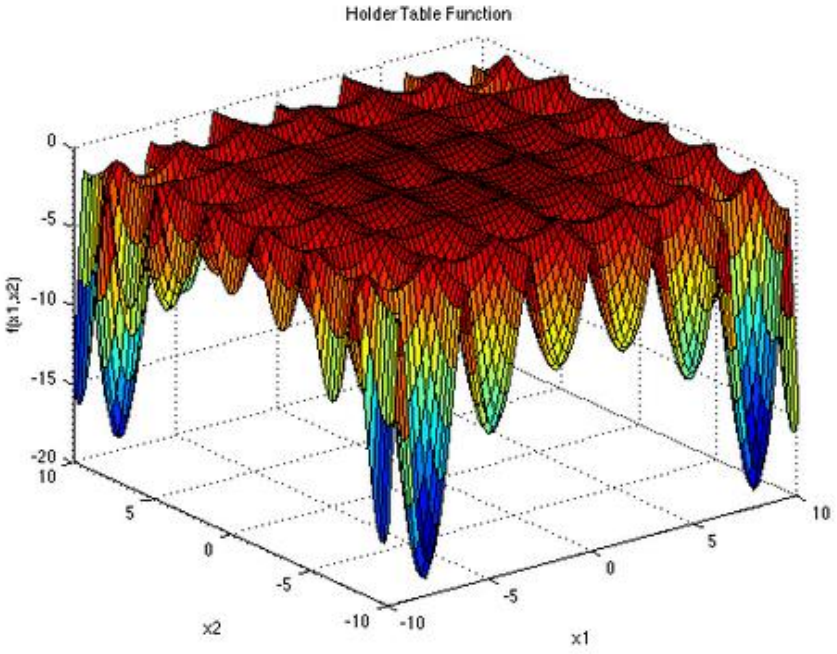
Генетический алгоритм:

d=10, количество нахождений глобального минимума 88%, число вычислений целевой функции не более 1250, максимальное значение 0,007251.

d=30, количество нахождений глобального минимума 83%, число вычислений целевой функции не более 1250, максимальное значение 0,074382.

d=50, количество нахождений глобального минимума 75%, число вычислений целевой функции не более 2500, максимальное значение 0,027493, для скрещивания отбиралось 40% популяции.

8. Функция Holder Table:



$$f(\mathbf{x}) = -\left| \sin(x_1)\cos(x_2)\exp\left(\left|1 - \frac{\sqrt{x_1^2 + x_2^2}}{\pi}\right|\right)\right|$$

Область определения:

$x_i \in [-10, 10]$, i = 1, 2.

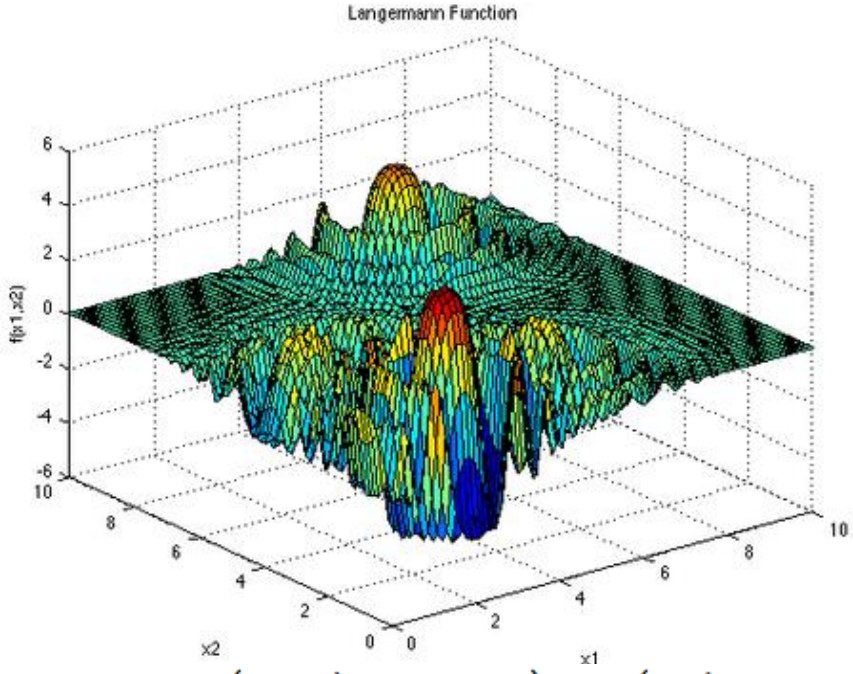
Глобальный минимум:

$f(\mathbf{x}^*) = -19.2085, \mathbf{x}^* = (8.05502, 9.66459), (8.05502, -9.66459), (-8.05502, 9.66459)$ и $(-8.05502, -9.66459)$

Генетический алгоритм:

d=2, количество нахождений глобального минимума 92%, число вычислений целевой функции не более 1250, максимальное значение -19,227325.

9. Функция Langermann:



$$f(\mathbf{x}) = \sum_{i=1}^{m} c_i \exp\left(-\frac{1}{\pi}\sum_{j=1}^{d}(x_j - A_{ij})^2\right)\cos\left(\pi\sum_{j=1}^{d}(x_j - A_{ij})^2\right)$$

Рекомендуемые переменные для d = 2: m = 5, c = (1, 2, 5, 2, 3) и

$$\mathbf{A} = \begin{pmatrix} 3 & 5 \\ 5 & 2 \\ 2 & 1 \\ 1 & 4 \\ 7 & 9 \end{pmatrix}$$

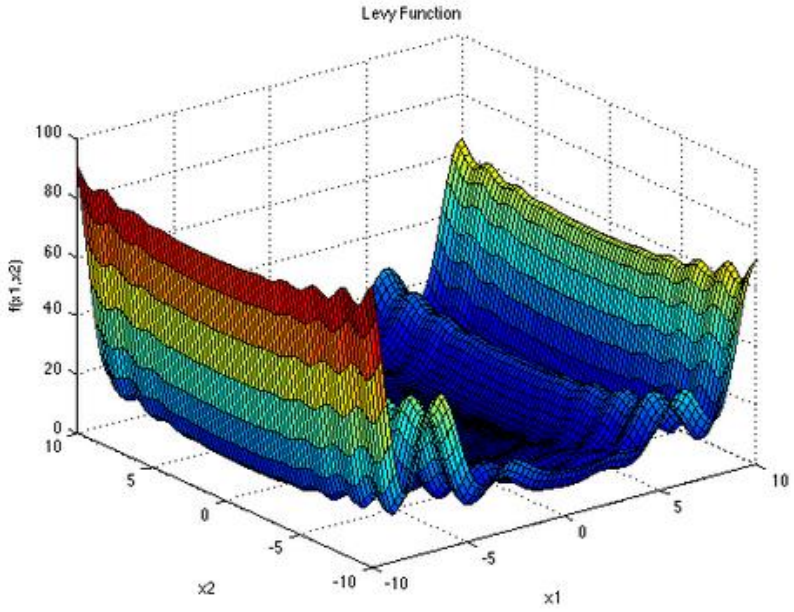
Область определения:

$x_i \in [0, 10]$, i = 1, …, d.

Генетический алгоритм:

d=2, количество нахождений глобального минимума 84%, число вычислений целевой функции не более 1250.

10. Функция Levy:



$$f(\mathbf{x}) = \sin^2(\pi w_1) + \sum_{i=1}^{d-1} (w_i - 1)^2 \left[1 + 10\sin^2(\pi w_i + 1)\right] + (w_d - 1)^2 \left[1 + \sin^2(2\pi w_d)\right],$$

$$w_i = 1 + \frac{x_i - 1}{4}, \quad i = 1, \ldots, d$$

Область определения:

$x_i \in [-10, 10]$, $i = 1, \ldots, d$.

Глобальный минимум:

$$f(\mathbf{x}^*) = 0, \quad \mathbf{x}^* = (1, \ldots, 1).$$

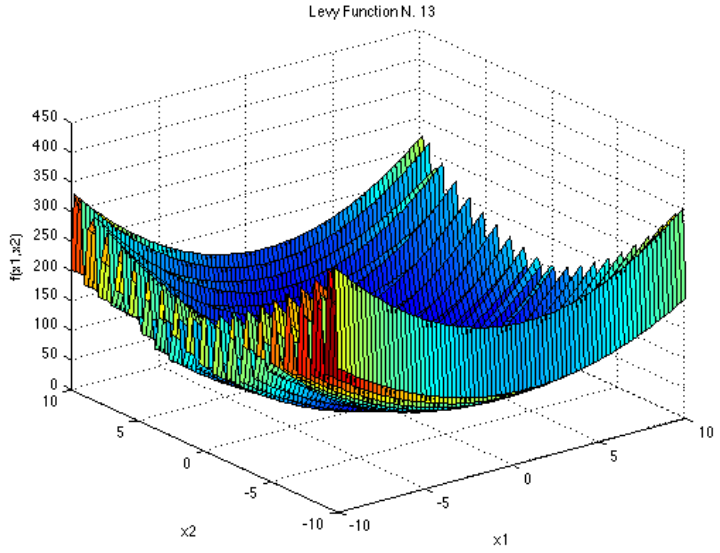
Генетический алгоритм:

d=10, количество нахождений глобального минимума 81%, число вычислений целевой функции не более 1250, максимальное значение 0,007437.

d=30, количество нахождений глобального минимума 79%, число вычислений целевой функции не более 1250, максимальное значение 0,073894.

d=50, количество нахождений глобального минимума 72%, число вычислений целевой функции не более 2500, максимальное значение 0,049284, для скрещивания отбиралось 40% популяции.

11. Функция Levy N. 13:



$$f(\mathbf{x}) = \sin^2(3\pi x_1) + (x_1 - 1)^2 \left[1 + \sin^2(3\pi x_2)\right] + (x_2 - 1)^2 \left[1 + \sin^2(2\pi x_2)\right]$$

Область определения:

$x_i \in [-10, 10]$, $i = 1, 2$.

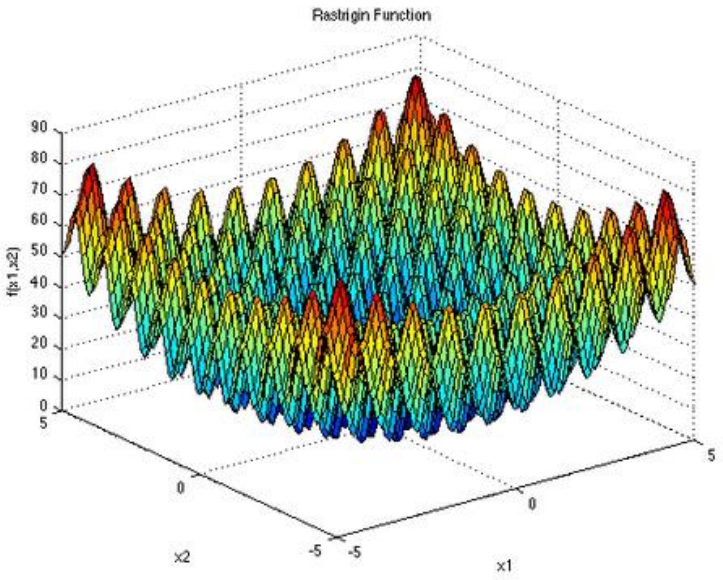
Глобальный минимум:

$$f(\mathbf{x}^*) = 0, \quad \mathbf{x}^* = (1, 1)$$

Генетический алгоритм:

d=2, количество нахождений глобального минимума 87%, число вычислений целевой функции не более 1250, максимальное значение 0,008325.

12. Функция Rastrigin:



$$f(\mathbf{x}) = 10d + \sum_{i=1}^{d} \left[x_i^2 - 10\cos(2\pi x_i)\right]$$

Область определения:

$x_i \in [-5.12, 5.12]$, $i = 1, \ldots, d$.

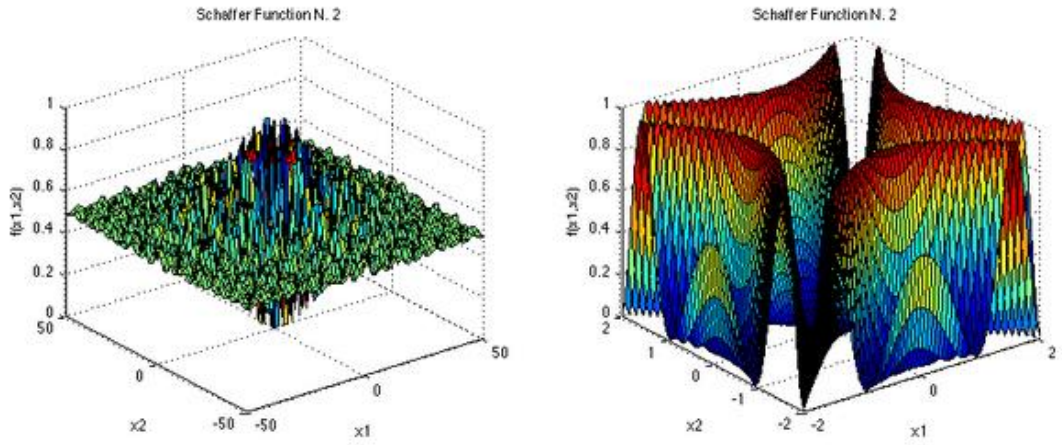
Глобальный минимум:

$$f(\mathbf{x}^*) = 0, \quad \mathbf{x}^* = (0, \ldots, 0)$$

Генетический алгоритм:

d=2 количество нахождений глобального минимума 85%, число вычислений целевой функции не более 1250, максимальное значение 0,007435.

13. Функция Schaffer N. 2:



$$f(\mathbf{x}) = 0.5 + \frac{\sin^2(x_1^2 - x_2^2) - 0.5}{\left[1 + 0.001(x_1^2 + x_2^2)\right]^2}$$

$x_i \in [-100, 100]$, $i = 1, 2$.

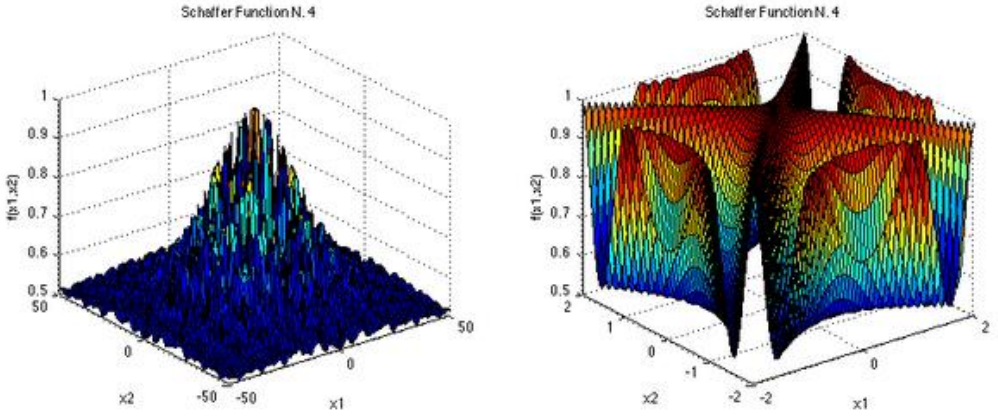
Глобальный минимум:

$$f(\mathbf{x}^*) = 0, \quad \mathbf{x}^* = (0, 0)$$

Генетический алгоритм:

d=2, количество нахождений глобального минимума 84%, число вычислений целевой функции не более 1250, максимальное значение 0,008325.

14. Функция Schaffer N. 4:



$$f(\mathbf{x}) = 0.5 + \frac{\cos\left(\sin(|x_1^2 - x_2^2|)\right) - 0.5}{\left[1 + 0.001(x_1^2 + x_2^2)\right]^2}$$

Область определения:

$x_i \in [-100, 100]$, $i = 1, 2$.

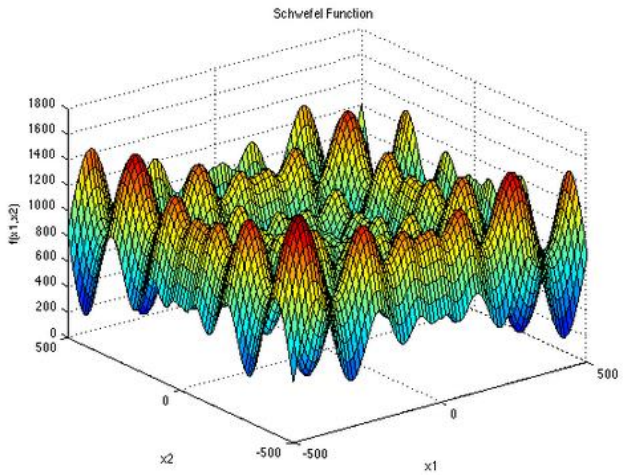
Глобальный минимум:

$$f(\mathbf{x}^*) = 0, \quad \mathbf{x}^* = (0, 0)$$

Генетический алгоритм:

d=2, количество нахождений глобального минимума 83%, число вычислений целевой функции не более 1250, максимальное значение 0,012743.

15. Функция Schwefel:



$$f(\mathbf{x}) = 418.9829d - \sum_{i=1}^{d} x_i \sin(\sqrt{|x_i|})$$

Область определения:

$x_i \in [-500, 500]$, $i = 1, \ldots, d$.

Глобальный минимум:

$$f(\mathbf{x}^*) = 0, \ \mathbf{x}^* = (420.9687, \ldots, 420.9687)$$

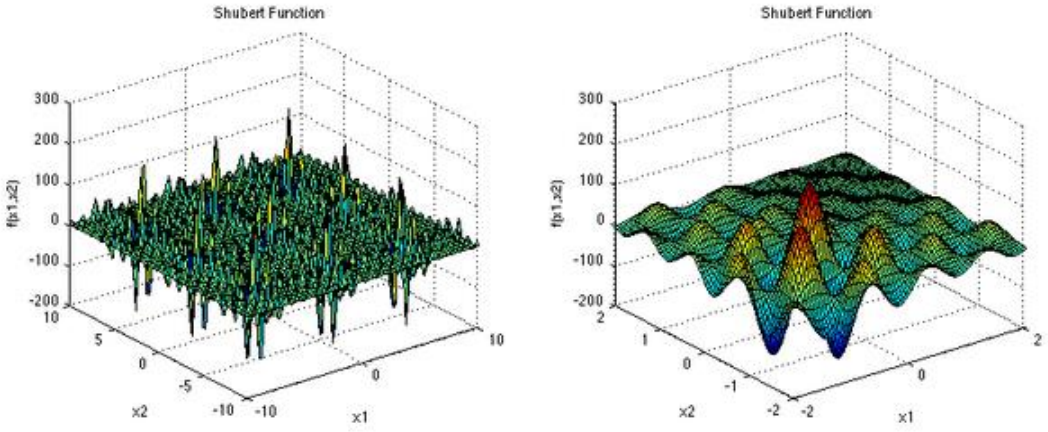
Генетический алгоритм:

d=10, количество нахождений глобального минимума 87%, число вычислений целевой функции не более 1250, максимальное значение 0,009324.

d=30, количество нахождений глобального минимума 74%, число вычислений целевой функции не более 1250, максимальное значение 0,093123.

d=50, количество нахождений глобального минимума 71%, число вычислений целевой функции не более 2500, максимальное значение 0,028375, для скрещивания отбиралось 40% популяции.

16. Функция Shubert:



$$f(\mathbf{x}) = \left(\sum_{i=1}^{5} i\cos((i+1)x_1 + i)\right)\left(\sum_{i=1}^{5} i\cos((i+1)x_2 + i)\right)$$

Область определения:

$x_i \in [-10, 10]$, $i = 1, 2$, $x_i \in [-5.12, 5.12]$, $i = 1, 2$.

Глобальный минимум:

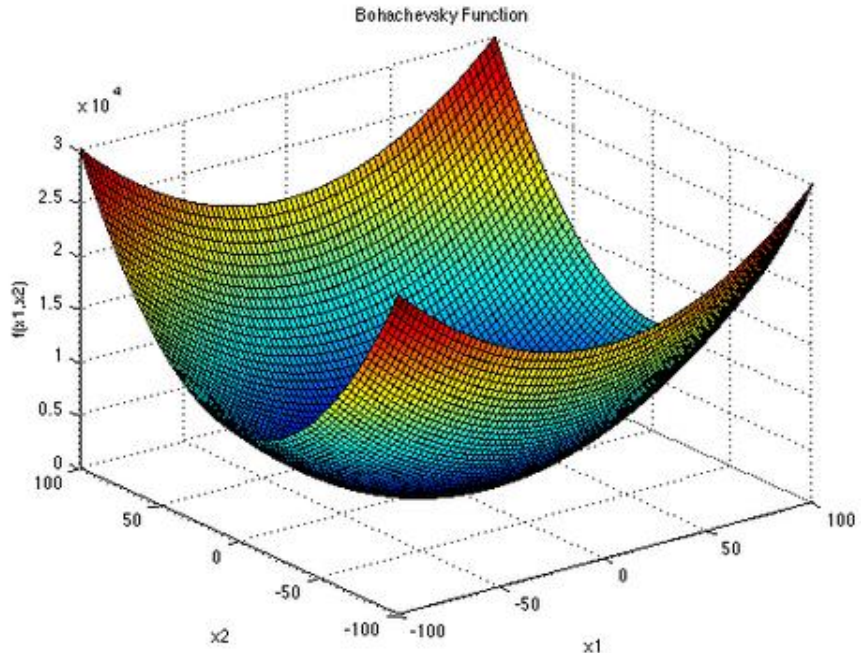
$$f(\mathbf{x}^*) = -186.7309$$

Генетический алгоритм:

d=2 количество нахождений глобального минимума 82%, число вычислений целевой функции не более 1250, максимальное значение 0,007392.

Чашеобразные функции

17. Функция Bohachevsky:



$$f_1(\mathbf{x}) = x_1^2 + 2x_2^2 - 0.3\cos(3\pi x_1) - 0.4\cos(4\pi x_2) + 0.7$$

$$f_2(\mathbf{x}) = x_1^2 + 2x_2^2 - 0.3\cos(3\pi x_1)\cos(4\pi x_2) + 0.3$$

$$f_3(\mathbf{x}) = x_1^2 + 2x_2^2 - 0.3\cos(3\pi x_1 + 4\pi x_2) + 0.3$$

Область определения:

$x_i \in [-100, 100]$, $i = 1, 2$.

Глобальный минимум:

$f_j(\mathbf{x}^*) = 0, \quad \mathbf{x}^* = (0, 0), \; j = 1, 2, 3$.

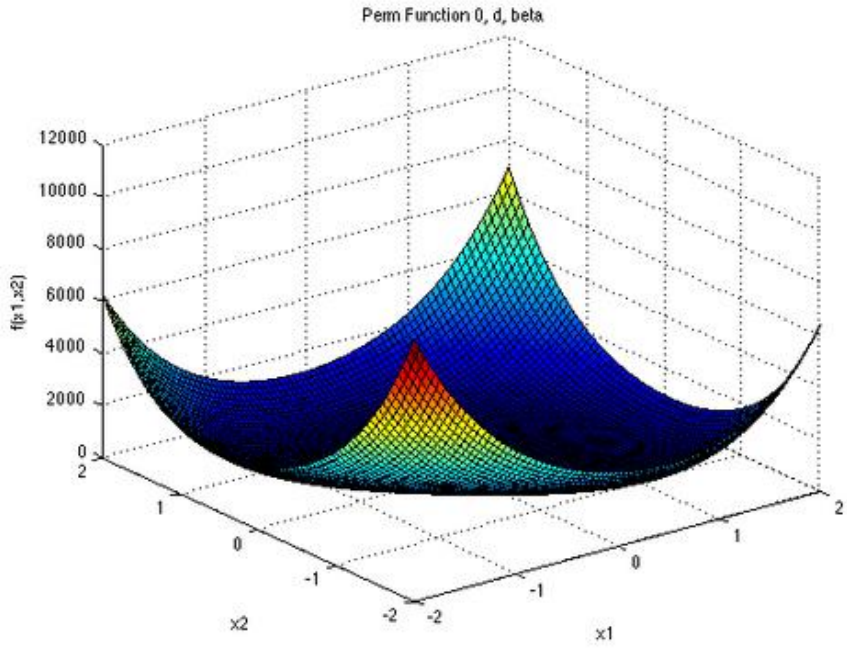
Генетический алгоритм:

j=1, количество нахождений глобального минимума 84%, число вычислений целевой функции не более 1250, максимальное значение 0,002956.

j=2, количество нахождений глобального минимума 87%, число вычислений целевой функции не более 1250, максимальное значение 0,005784.

j=3, количество нахождений глобального минимума 75%, число вычислений целевой функции не более 1250, максимальное значение 0,018465, для скрещивания отбиралось 40% популяции.

18. Функция Perm 0, d, beta:



$$f(\mathbf{x}) = \sum_{i=1}^{d} \left(\sum_{j=1}^{d} (j+\beta) \left(x_j^i - \frac{1}{j^i} \right) \right)^2$$

Область определения:

$x_i \in [-d, d]$, $i = 1, \ldots, d$.

Глобальный минимум:

$f(\mathbf{x}^*) = 0,\ \mathbf{x}^* = \left(1, \frac{1}{2}, \ldots, \frac{1}{d}\right)$

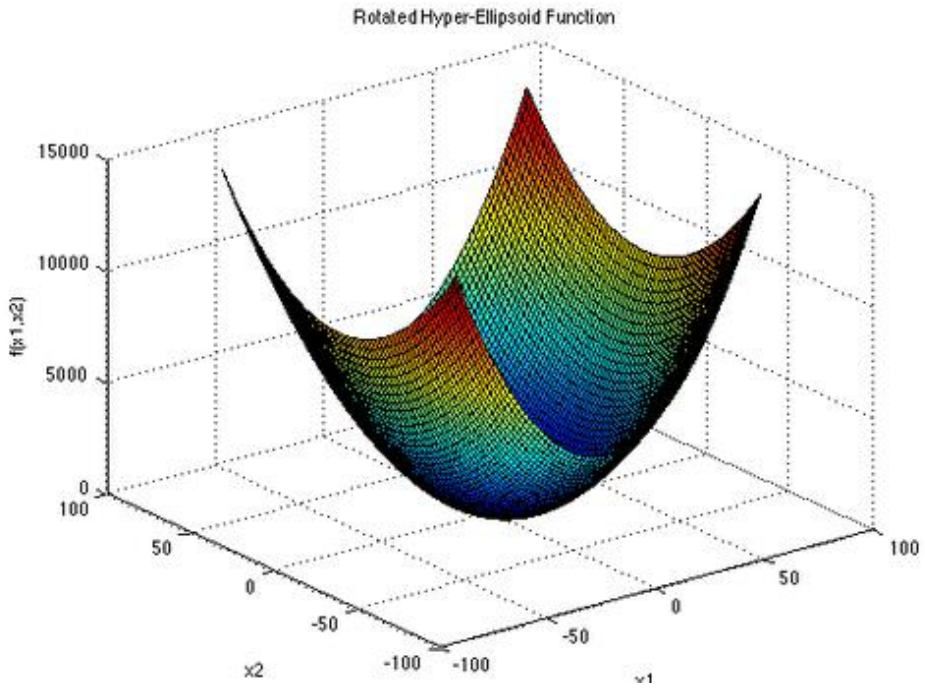
Генетический алгоритм:

d=10, количество нахождений глобального минимума 89%, число вычислений целевой функции не более 1250, максимальное значение 0,007859.

d=30, количество нахождений глобального минимума 84%, число вычислений целевой функции не более 1250, максимальное значение 0,028647.

d=50, количество нахождений глобального минимума 75%, число вычислений целевой функции не более 2500, максимальное значение 0,089542, для скрещивания отбиралось 40% популяции.

19. Функция Rotated Hyper-Ellipsoid:



$$f(\mathbf{x}) = \sum_{i=1}^{d} \sum_{j=1}^{i} x_j^2$$

Область определения:

$x_i \in [-65.536, 65.536]$, $i = 1, \ldots, d$.

Глобальный минимум:

$$f(\mathbf{x}^*) = 0,\ \mathbf{x}^* = (0, \ldots, 0)$$

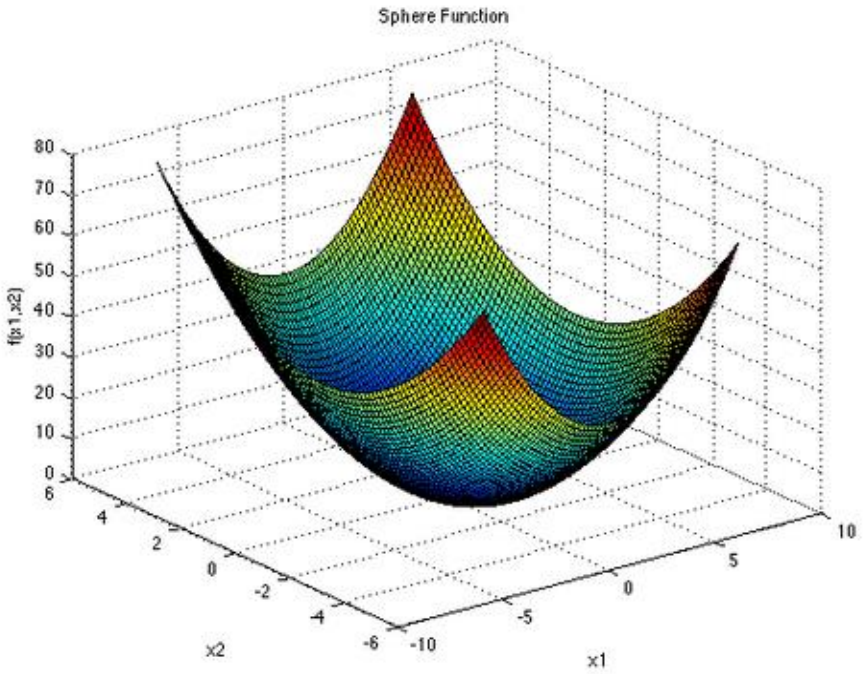
Генетический алгоритм:

d=10, количество нахождений глобального минимума 87%, число вычислений целевой функции не более 1250, максимальное значение 0,004895.

d=30, количество нахождений глобального минимума 74%, число вычислений целевой функции не более 1250, максимальное значение 0,074835.

d=50, количество нахождений глобального минимума 71%, число вычислений целевой функции не более 2500, максимальное значение 0,028347, для скрещивания отбиралось 40% популяции.

20. Функция Sphere:



$$f(\mathbf{x}) = \sum_{i=1}^{d} x_i^2$$

Область определения:

$x_i \in [-5.12, 5.12]$, $i = 1, \ldots, d$.

Глобальный минимум:

$$f(\mathbf{x}^*) = 0, \mathbf{x}^* = (0, \ldots, 0)$$

Генетический алгоритм:

d=10, количество нахождений глобального минимума (вообще-то он здесь один...:)) 86%, число вычислений целевой функции не более 1250, максимальное значение 0,008325.

d=30, количество нахождений глобального минимума 74%, число вычислений целевой функции не более 1250, максимальное значение 0,108851.

d=50, количество нахождений глобального минимума 56%, число вычислений целевой функции не более 2500, максимальное значение 0,016291, для скрещивания отбиралось 40% популяции.

Модифицированная форма:

d = 6, в $[0, 1]^6$:

$$f(\mathbf{x}) = \frac{1}{899}\left(\sum_{i=1}^{6} x_i^2 2^i - 1745\right).$$

21. Функция Sum of Different Powers:

$$f(\mathbf{x}) = \sum_{i=1}^{d} |x_i|^{i+1}$$

Область определения:

$x_i \in [-1, 1]$, $i = 1, \ldots, d$.

Глобальный минимум:

$f(\mathbf{x}^*) = 0,\ \mathbf{x}^* = (0, \ldots, 0)$.

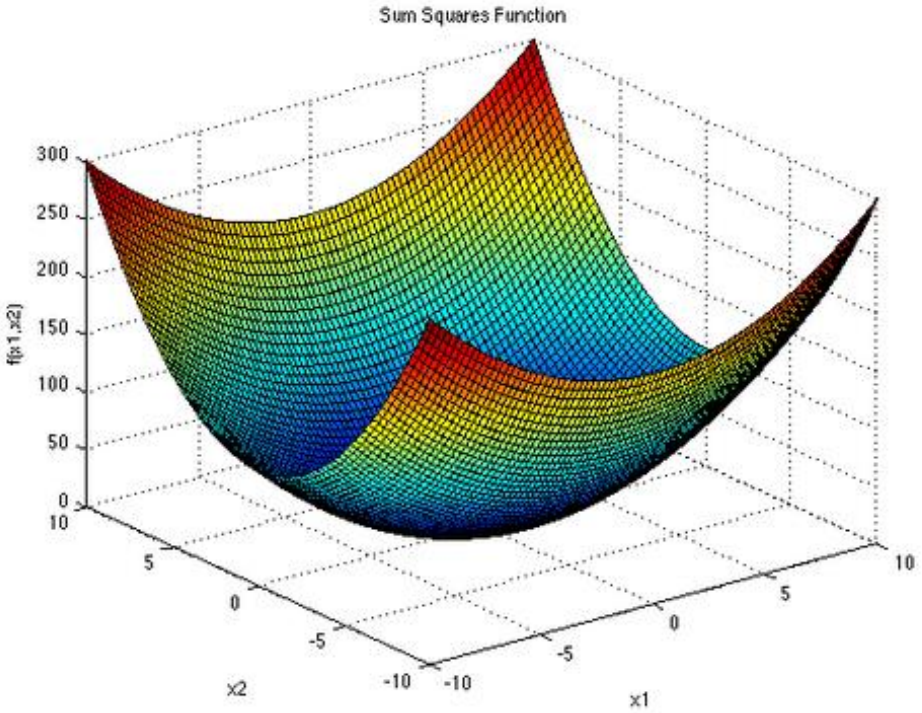
Генетический алгоритм:

d=10, количество нахождений глобального минимума 85%, число вычислений целевой функции не более 1250, максимальное значение 0,006447.

d=30, количество нахождений глобального минимума 71%, число вычислений целевой функции не более 1250, максимальное значение 0,038646.

d=50, количество нахождений глобального минимума 69%, число вычислений целевой функции не более 2500, максимальное значение 0,095368.

22. Функция Sum Squares:



$$f(\mathbf{x}) = \sum_{i=1}^{d} i x_i^2$$

Область определения:

$x_i \in [-10, 10]$, i = 1, …, d, $x_i \in [-5.12, 5.12]$, i = 1, …, *d*.

Глобальный минимум:

$f(\mathbf{x}^*) = 0\ ,\ \mathbf{x}^* = (0, \ldots, 0)$

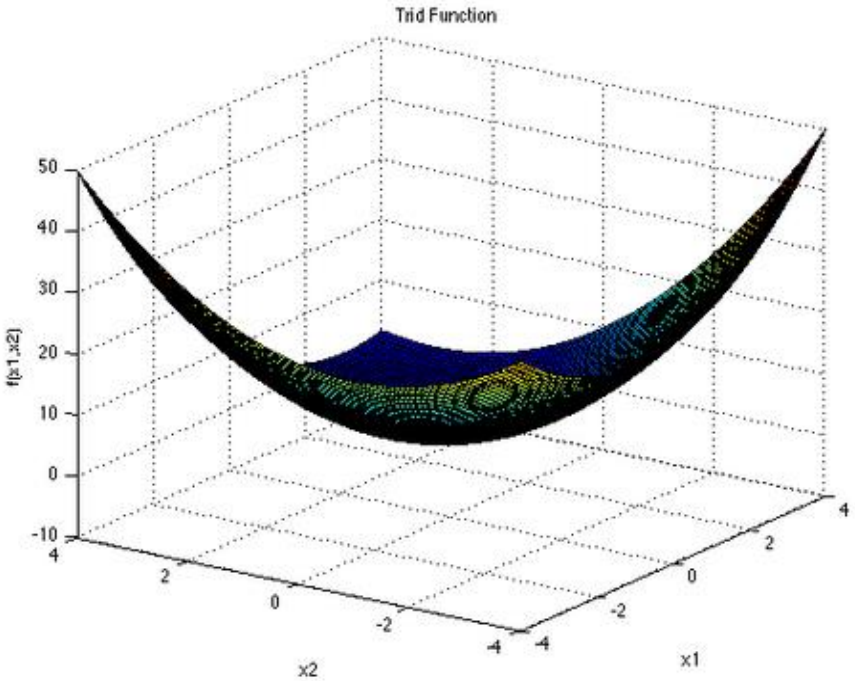
Генетический алгоритм:

d=10, количество нахождений глобального минимума 87%, число вычислений целевой функции не более 1250, максимальное значение 0,004895.

d=30, количество нахождений глобального минимума 74%, число вычислений целевой функции не более 1250, максимальное значение 0,074835.

d=50, количество нахождений глобального минимума 71%, число вычислений целевой функции не более 2500, максимальное значение 0,028347, для скрещивания отбиралось 40% популяции.

23. Функция Trid Function:



$$f(\mathbf{x}) = \sum_{i=1}^{d} (x_i - 1)^2 - \sum_{i=2}^{d} x_i x_{i-1}$$

Область определения:

$x_i \in [-d^2, d^2]$, i = 1, …, *d*.

Глобальный минимум:

$d = 6$: $f(\mathbf{x}^*) = -50$

$d = 10$: $f(\mathbf{x}^*) = -200$

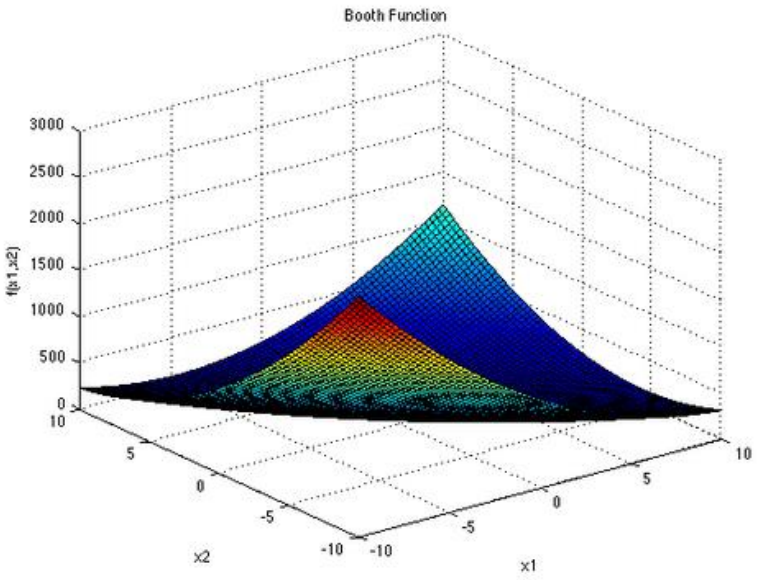
Генетический алгоритм:

d=6, количество нахождений глобального минимума 87%, число вычислений целевой функции не более 1250, максимальное значение -50,009564.

d=10, количество нахождений глобального минимума 74%, число вычислений целевой функции не более 1250, максимальное значение -200,058349.

Пластинчатые функции:

24. Функция Booth Function:



$$f(\mathbf{x}) = (x_1 + 2x_2 - 7)^2 + (2x_1 + x_2 - 5)^2$$

Область определения:

$x_i \in [-10, 10]$, $i = 1, 2$.

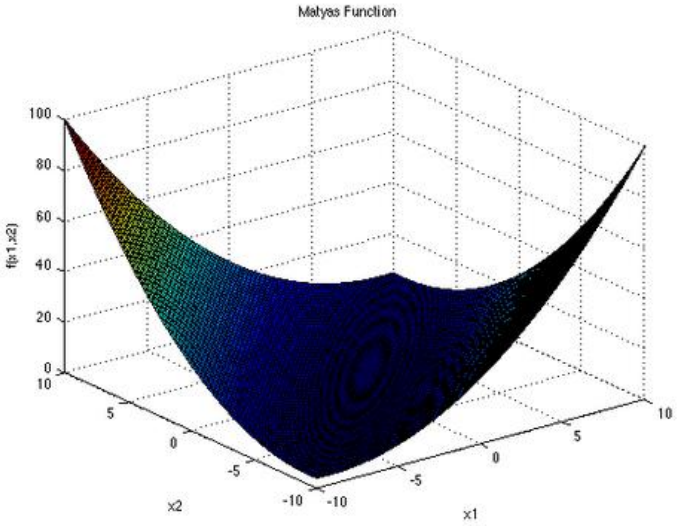
Глобальный минимум:

$f(\mathbf{x}^*) = 0, \ \mathbf{x}^* = (1, 3)$

Генетический алгоритм:

d=2, количество нахождений глобального минимума 91%, число вычислений целевой функции не более 1250, максимальное значение 0,006795.

25. Функция Matyas:



$$f(\mathbf{x}) = 0.26(x_1^2 + x_2^2) - 0.48x_1x_2$$

Область определения:

$x_i \in [-10, 10]$, i = 1, 2.

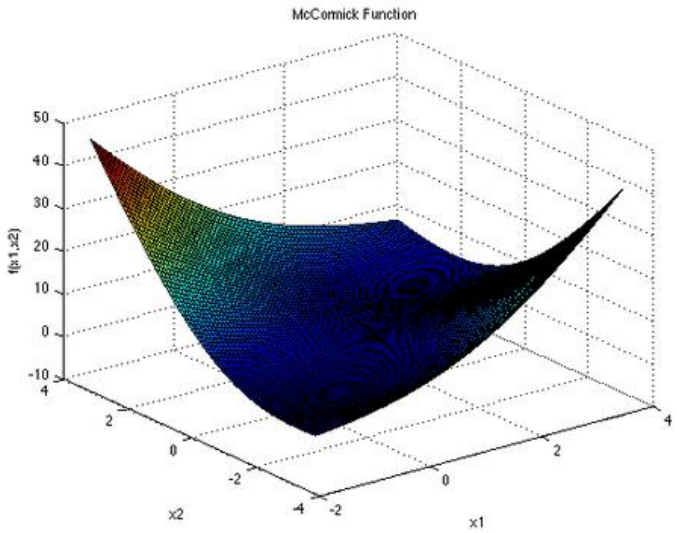
Глобальный минимум:

$f(\mathbf{x}^*) = 0$, at $\mathbf{x}^* = (0, 0)$ $f(\mathbf{x}^*) = 0$, at $\mathbf{x}^* = (0, 0)$

Генетический алгоритм:

d=2, количество нахождений глобального минимума 77%, число вычислений целевой функции не более 1250, максимальное значение 0,008794.

26. Функция McCormick:



$$f(\mathbf{x}) = \sin(x_1 + x_2) + (x_1 - x_2)^2 - 1.5x_1 + 2.5x_2 + 1$$

Область определения:

$x_1 \in [-1.5, 4]$, $x_2 \in [-3, 4]$.

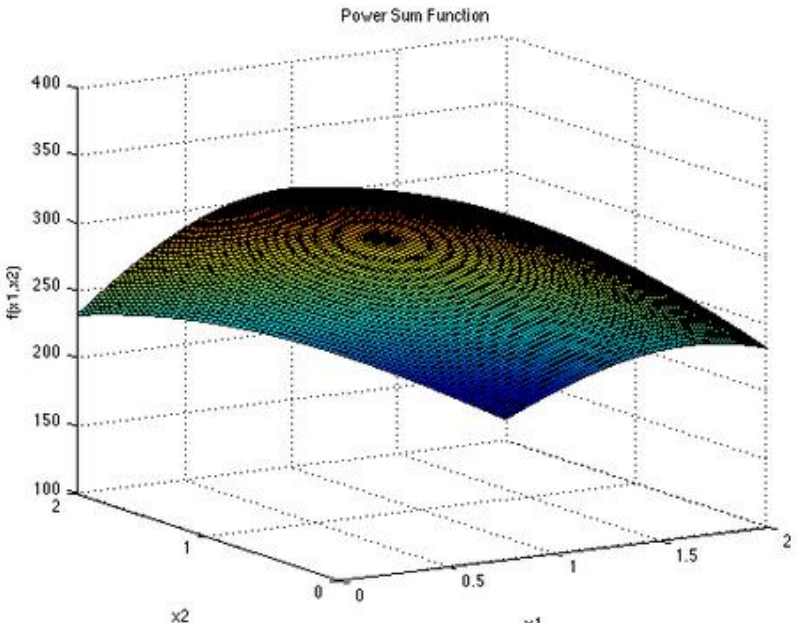
Глобальный минимум:

$$f(\mathbf{x}^*) = -1.9133, \quad \mathbf{x}^* = (-0.54719, -1.54719)$$

Генетический алгоритм:

d=2, количество нахождений глобального минимума 87%, число вычислений целевой функции не более 1250, максимальное значение 1,875439.

27. Функция Power Sum:



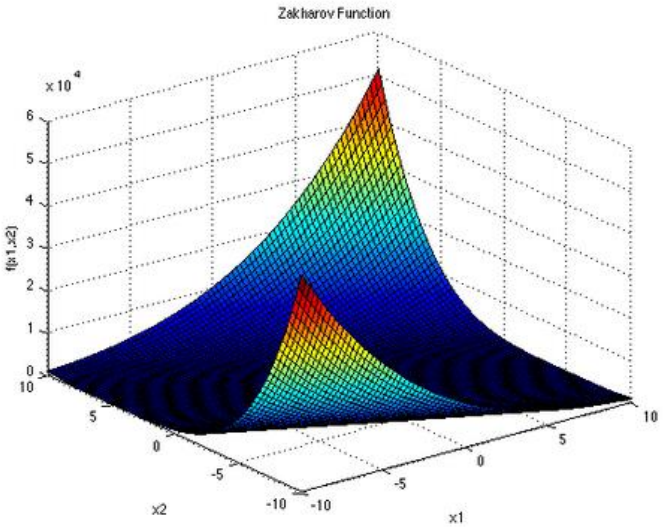
$$f(\mathbf{x}) = \sum_{i=1}^{d} \left[\left(\sum_{j=1}^{d} x_j^i \right) - b_i \right]^2$$

Рекомендуемые переменные для *d = 4: b = (8, 18, 44, 114).*

Область определения:

$x_i \in [0, d]$, i = 1, …, *d*.

28. Функция Zakharov:



$$f(\mathbf{x}) = \sum_{i=1}^{d} x_i^2 + \left(\sum_{i=1}^{d} 0.5ix_i\right)^2 + \left(\sum_{i=1}^{d} 0.5ix_i\right)^4$$

Область определения:

$x_i \in [-5, 10]$, $i = 1, \ldots, d$.

Глобальный минимум:

$f(\mathbf{x}^*) = 0, \quad \mathbf{x}^* = (0, \ldots, 0)$

Генетический алгоритм:

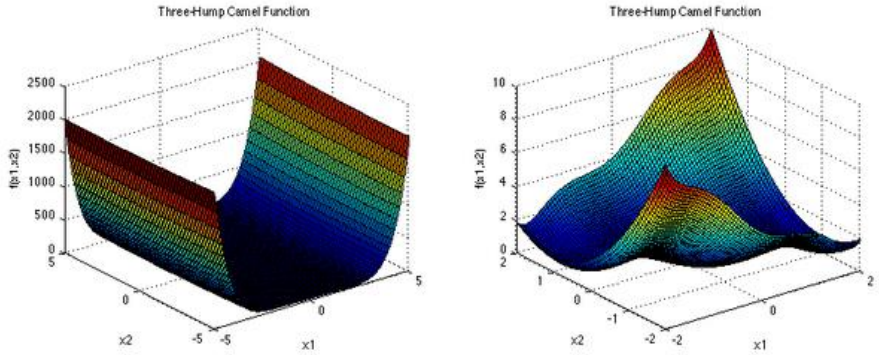
d=10, количество нахождений глобального минимума 91%, число вычислений целевой функции не более 1250, максимальное значение 0,003946.

d=30, количество нахождений глобального минимума 83%, число вычислений целевой функции не более 1250, максимальное значение 0,075963.

d=50, количество нахождений глобального минимума 78%, число вычислений целевой функции не более 2500, максимальное значение 0,059485, для скрещивания отбиралось 40% популяции.

Долина-образные функции:

29. Функция Three-Hump Camel:



$$f(\mathbf{x}) = 2x_1^2 - 1.05x_1^4 + \frac{x_1^6}{6} + x_1x_2 + x_2^2$$

Область определения:

$x_i \in [-5, 5]$, $i = 1, 2$.

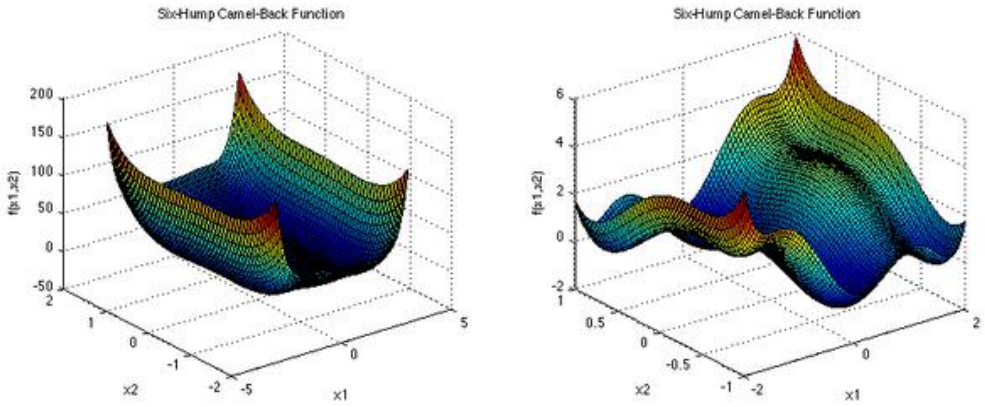
Глобальный минимум:

$$f(\mathbf{x}^*) = 0, \ \mathbf{x}^* = (0, 0)$$

Генетический алгоритм:

d=2, количество нахождений глобального минимума 83%, число вычислений целевой функции не более 1250, максимальное значение 0,004857.

30. Функция Six-Hump Camel:



$$f(\mathbf{x}) = \left(4 - 2.1x_1^2 + \frac{x_1^4}{3}\right)x_1^2 + x_1x_2 + (-4 + 4x_2^2)x_2^2$$

Область определения:

$x_1 \in [-3, 3]$, $x_2 \in [-2, 2]$.

Глобальный минимум:

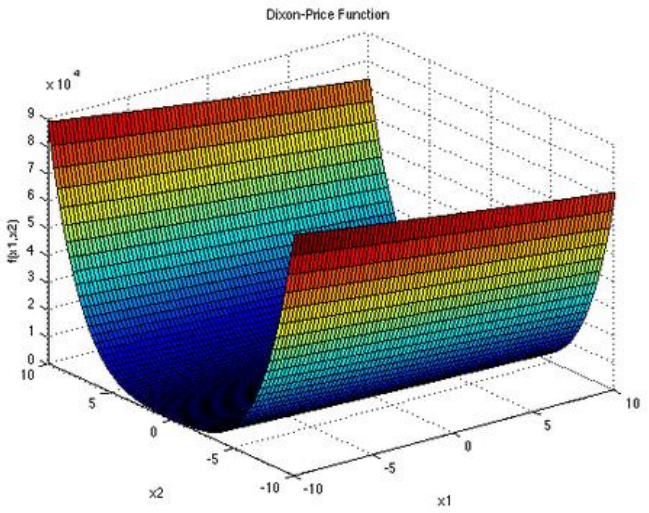
$f(\mathbf{x}^*) = -1.0316,$

$\mathbf{x}^* = (0.0898, -0.7126)$ и $(-0.0898, 0.7126)$

Генетический алгоритм:

d=2, количество нахождений глобального минимума 86%, число вычислений целевой функции не более 1250, максимальное значение -1,03575.

31. Функция Dixon-Price:



$$f(\mathbf{x}) = (x_1 - 1)^2 + \sum_{i=2}^{d} i \left(2x_i^2 - x_{i-1}\right)^2$$

Область определения:

$x_i \in [-10, 10]$, $i = 1, \ldots, d$.

Глобальный минимум:

$f(\mathbf{x}^*) = 0, \quad x_i = 2^{-\frac{2^i - 2}{2^i}}$

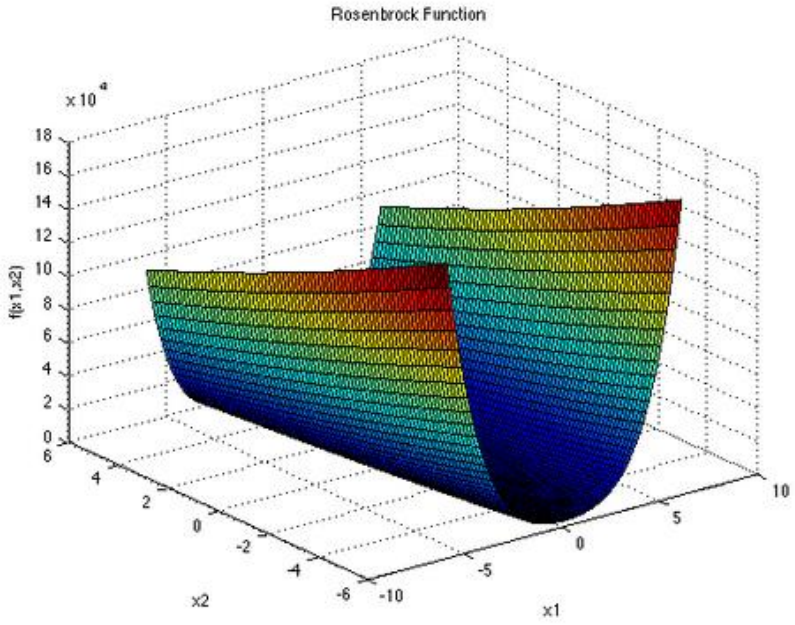
Генетический алгоритм:

d=10, количество нахождений глобального минимума 82%, число вычислений целевой функции не более 1250, максимальное значение 0,002834.

d=30, количество нахождений глобального минимума 75%, число вычислений целевой функции не более 1250, максимальное значение 0,029567.

d=50, количество нахождений глобального минимума 57%, число вычислений целевой функции не более 2500, максимальное значение 0,075846, для скрещивания отбиралось 40% популяции.

32. Функция Rosenbrock:



$$f(\mathbf{x}) = \sum_{i=1}^{d-1} \left[100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2\right]$$

Область определения:

$x_i \in [-5, 10]$, i = 1, …, d, $x_i \in [-2.048, 2.048]$, i = 1, …, *d*.

Глобальный минимум:

$f(\mathbf{x}^*) = 0,\ \mathbf{x}^* = (1, \ldots, 1)$

Симплекс метод Нелдера Мида:

xmin = 1.0000; 1.0000, opt = 4.1940e-014.

Метод градиента: xmin = 1.0000; 1.0000, opt = 1.9116e-011.

xmin = 0.6120; 0.3715, opt = 0.1446.

Обратите внимание на то, что вопреки ожиданиям функция к успеху не привела.

Генетический алгоритм:

d=2, количество нахождений глобального минимума 92%, число вычислений целевой функции не более 1250, максимальное значение функции 0,00169118.

d=4, количество нахождений глобального минимума 86%, число вычислений целевой функции не более 1250, максимальное значение функции 0,00504982.

Крутые восхождения / снижения.

33. Функция De Jong N. 5:

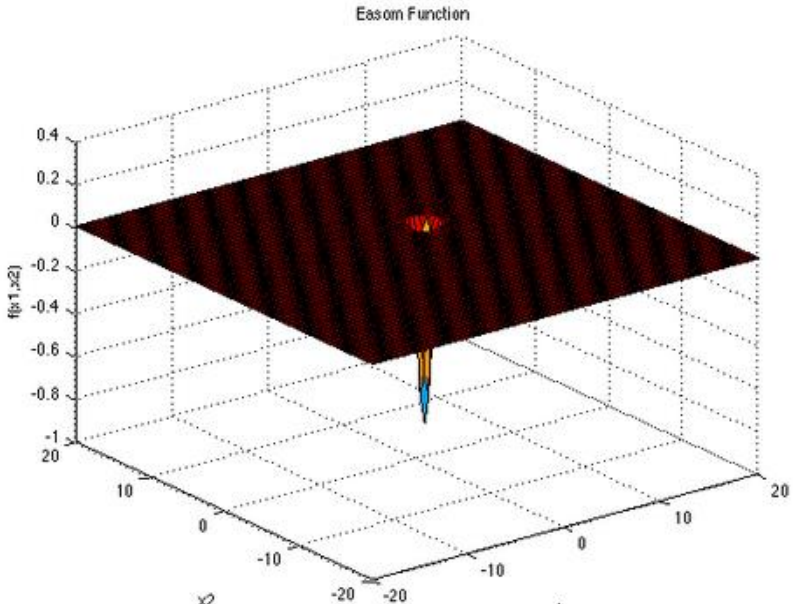
$$f(\mathbf{x}) = \left(0.002 + \sum_{i=1}^{25} \frac{1}{i + (x_1 - a_{1i})^6 + (x_2 - a_{2i})^6} \right)^{-1}$$

$$\mathbf{a} = \begin{pmatrix} -32 & -16 & 0 & 16 & 32 & -32 & \dots & 9 & 16 & 32 \\ -32 & -32 & -32 & -32 & -32 & -16 & \dots & 32 & 32 & 32 \end{pmatrix}$$

Область определения:

$x_i \in [-65.536, 65.536]$, i = 1, 2.

34. Функция Easom:



$$f(\mathbf{x}) = -\cos(x_1)\cos(x_2)\exp\left(-(x_1 - \pi)^2 - (x_2 - \pi)^2\right)$$

Область определения:

$x_i \in [-100, 100]$, i = 1, 2.

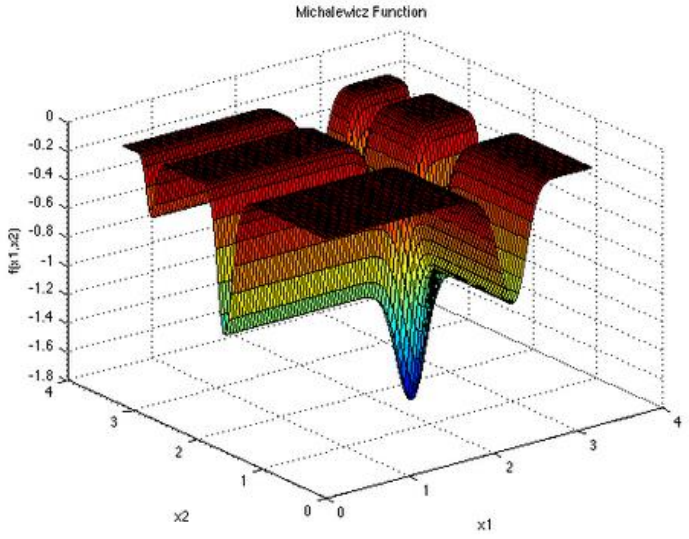
Глобальный минимум:

$f(\mathbf{x}^*) = -1$, at $\mathbf{x}^* = (\pi, \pi)$

Генетический алгоритм:

d=2 количество нахождений глобального минимума 87 число вычислений целевой функции не более 1250, максимальное значение -0,990463.

35. Функция Michalewicz :



$$f(\mathbf{x}) = -\sum_{i=1}^{d} \sin(x_i)\sin^{2m}\left(\frac{ix_i^2}{\pi}\right)$$

Рекомендуемые переменные m = 10.

Область определения:

$x_i \in [0, \pi]$, i = 1, …, d.

Глобальный минимум:

$d = 2$: $f(\mathbf{x}^*) = -1.8013$, $\mathbf{x}^* = (2.20, 1.57)$

$d = 5$: $f(\mathbf{x}^*) = -4.687658$

$d = 10$: $f(\mathbf{x}^*) = -9.66015$

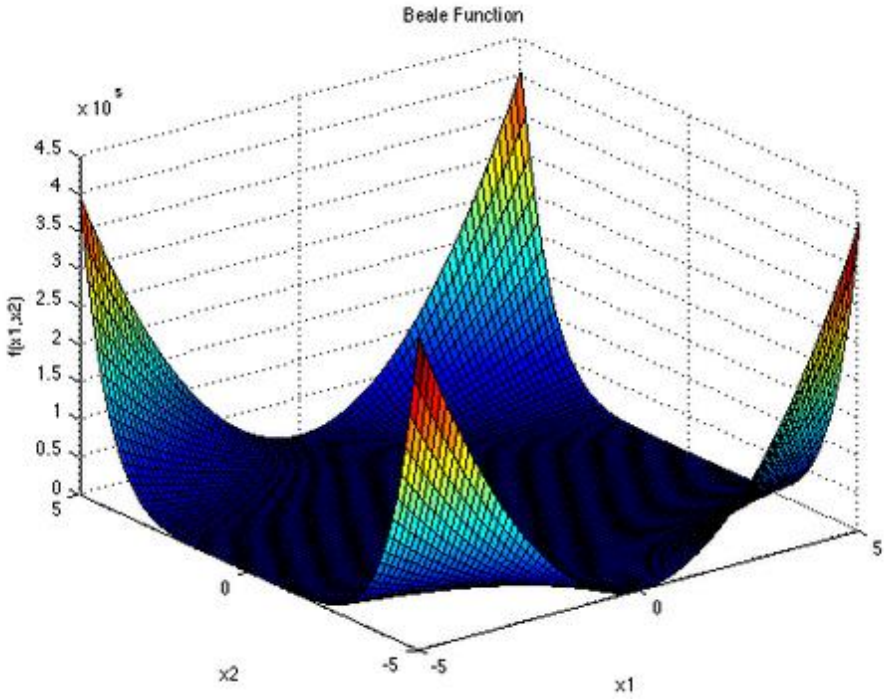
Генетический алгоритм:

d=2 количество нахождений глобального минимума 88 число вычислений целевой функции не более 1250, максимальное значение -1,799871

d=5 количество нахождений глобального минимума 79 число вычислений целевой функции не более 1250, максимальное значение -4,675963.

d=10, количество нахождений глобального минимума 75%, число вычислений целевой функции не более 2500, максимальное значение -9, 650594, для скрещивания отбиралось 40% популяции.

36. Функция Beale:



$$f(\mathbf{x}) = (1.5 - x_1 + x_1 x_2)^2 + (2.25 - x_1 + x_1 x_2^2)^2 + (2.625 - x_1 + x_1 x_2^3)^2$$

Область определения:

$x_i \in [-4.5, 4.5]$, $i = 1, 2$.

Глобальный минимум:

$$f(\mathbf{x}^*) = 0,\ \mathbf{x}^* = (3, 0.5)$$

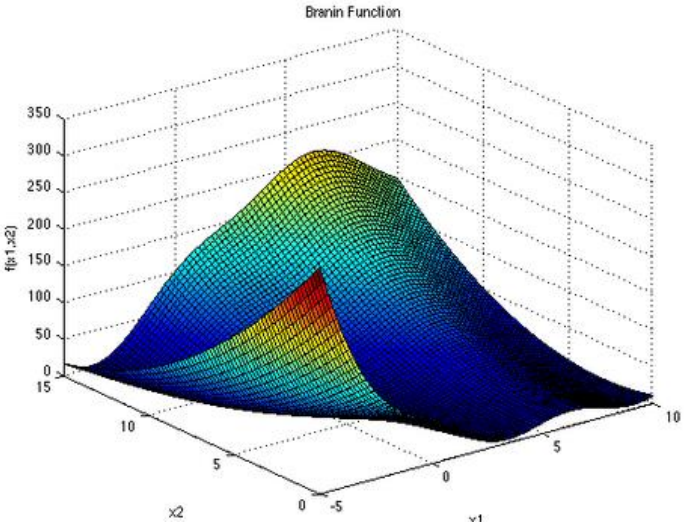
Генетический алгоритм:

d=10, количество нахождений глобального минимума 81%, число вычислений целевой функции не более 1250, максимальное значение 0,004586.

d=30, количество нахождений глобального минимума 73%, число вычислений целевой функции не более 1250, максимальное значение 0,047354.

d=50, количество нахождений глобального минимума 65%, число вычислений целевой функции не более 2500, максимальное значение 0,049386, для скрещивания отбиралось 40% популяции.

37. Функция Branin:



$$f(\mathbf{x}) = a(x_2 - bx_1^2 + cx_1 - r)^2 + s(1-t)\cos(x_1) + s$$

Рекомендуемые переменные: a = 1, b = $5.1 / (4\pi^2)$, c = $5 / \pi$, r = 6, s = 10 и t = $1 / (8\pi)$.

Область определения:

$x_1 \in [-5, 10]$, $x_2 \in [0, 15]$.

Глобальный минимум:

$$f(\mathbf{x}^*) = 0.397887,$$

$$\mathbf{x}^* = (-\pi, 12.275), (\pi, 2.275) \text{ и } (9.42478, 2.475)$$

Генетический алгоритм:

d=2, количество нахождений глобального минимума 82%, число вычислений целевой функции не более 1250, максимальное значение 0,408574.

38. Функция Lville:

$$f(\mathbf{x}) = 100(x_1^2 - x_2)^2 + (x_1 - 1)^2 + (x_3 - 1)^2 + 90(x_3^2 - x_4)^2 + 10.1((x_2 - 1)^2 + (x_4 - 1)^2) + 19.8(x_2 - 1)(x_4 - 1)$$

Область определения:

$x_i \in [-10, 10]$, i = 1, 2, 3, 4.

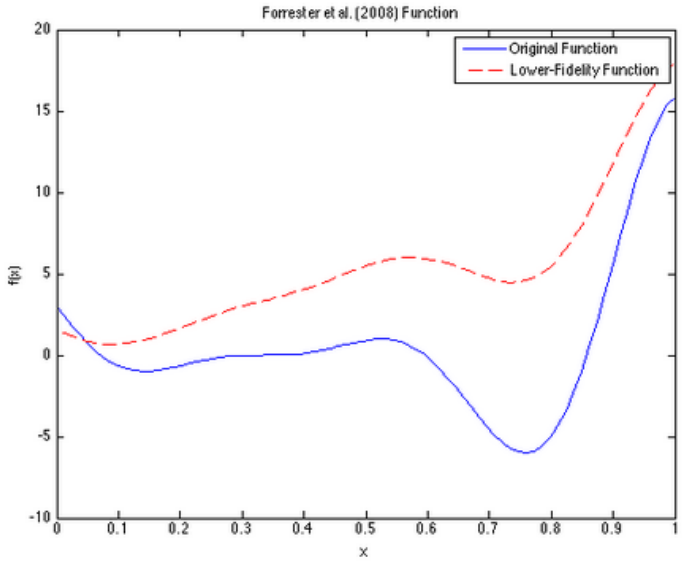
Глобальный минимум:

$$f(\mathbf{x}^*) = 0, \ \mathbf{x}^* = (1, 1, 1, 1)$$

Генетический алгоритм:

d=2, количество нахождений глобального минимума 78%, число вычислений целевой функции не более 1250, максимальное значение 0,004837.

39. Функция Forrester et al. (2008):



$$f(x) = (6x - 2)^2 \sin(12x - 4)$$

Область определения :

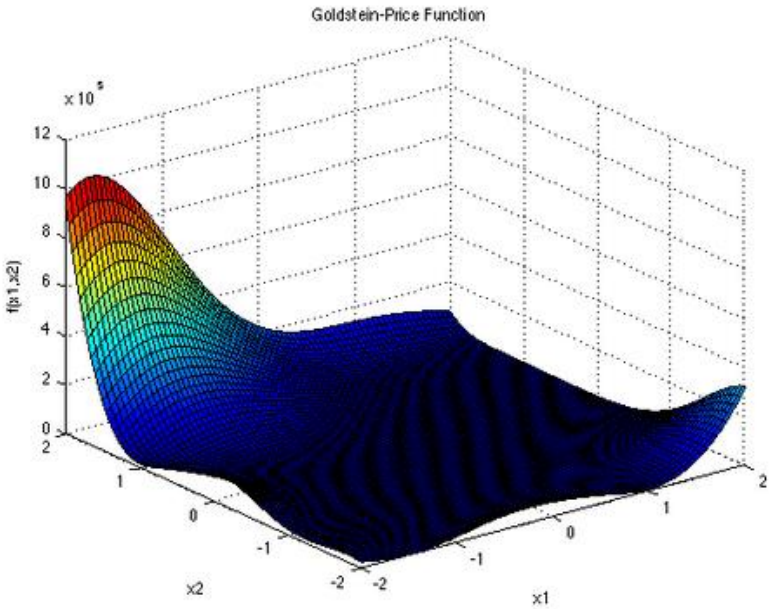
x ∈ [0, 1].

Модифицированная форма:

$$f_L(x) = Af(x) + B(x - 0.5) - C$$

Рекомендуемые параметры: A = 0.5, B = 10, C = -5.

40. Функция Goldstein-Price :



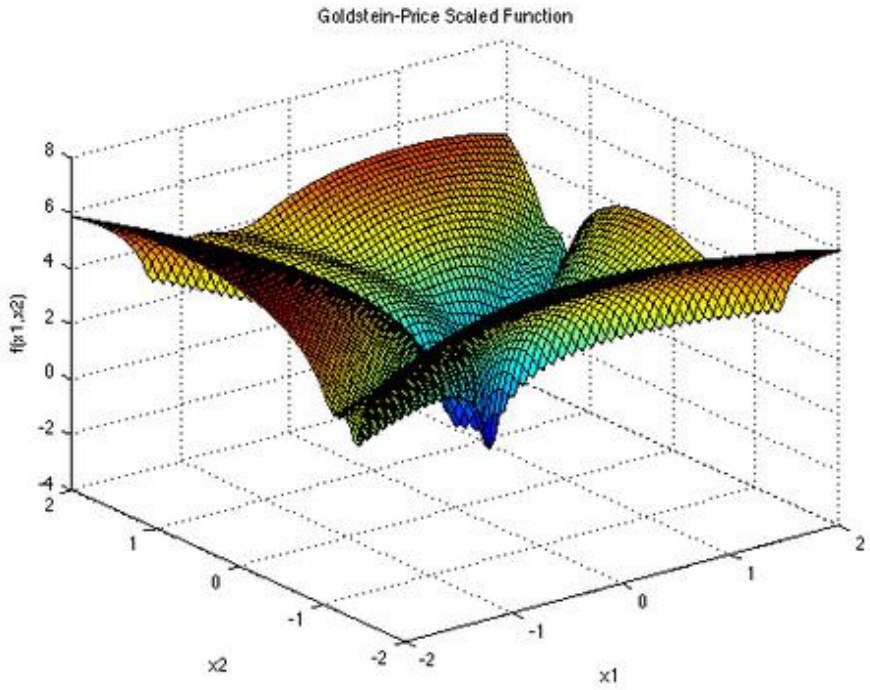
$$f(\mathbf{x}) = \left[1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)\right] \times \left[30 + (2x_1 - 3x_2)^2(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)\right]$$

Область определения:

$x_i \in [-2, 2]$, $i = 1, 2$.

Глобальный минимум:

$$f(\mathbf{x}^*) = 3, \quad \mathbf{x}^* = (0, -1)$$



$$f(\mathbf{x}) = \frac{1}{2.427}\left[\log\left(\left[1 + (\bar{x}_1 + \bar{x}_2 + 1)^2 \left(19 - 14\bar{x}_1 + 3\bar{x}_1^2 - 14\bar{x}_2 + 6\bar{x}_1\bar{x}_2 + 3\bar{x}_2^2\right)\right]\right.\right.$$
$$\left.\left.\left[30 + (2\bar{x}_1 - 3\bar{x}_2)^2 \left(18 - 32\bar{x}_1 + 12\bar{x}_1^2 + 48\bar{x}_2 - 36\bar{x}_1\bar{x}_2 + 27\bar{x}_2^2\right)\right]\right) - 8.693\right],$$

где

$$\bar{x}_i = 4x_i - 2, \; i = 1, 2$$

.

41. Функция Hartmann 3-Dimensional:

$$f(\mathbf{x}) = -\sum_{i=1}^{4} \alpha_i \exp\left(-\sum_{j=1}^{3} A_{ij}(x_j - P_{ij})^2\right),$$

где

$$\alpha = (1.0, 1.2, 3.0, 3.2)^T$$

$$\mathbf{A} = \begin{pmatrix} 3.0 & 10 & 30 \\ 0.1 & 10 & 35 \\ 3.0 & 10 & 30 \\ 0.1 & 10 & 36 \end{pmatrix}$$

$$\mathbf{P} = 10^{-4} \begin{pmatrix} 3689 & 1170 & 2673 \\ 4699 & 4387 & 7470 \\ 1091 & 8732 & 5547 \\ 381 & 5743 & 8828 \end{pmatrix}$$

Область определения:

$x_i \in (0, 1)$, i = 1, 2, 3.

Глобальный минимум:

$$f(\mathbf{x}^*) = -3.86278, \ \mathbf{x}^* = (0.114614, 0.555649, 0.852547)$$

42. Функция Hartmann 4-Dimensional:

$$f(\mathbf{x}) = \frac{1}{0.839} \left[1.1 - \sum_{i=1}^{4} \alpha_i \exp\left(- \sum_{j=1}^{4} A_{ij} (x_j - P_{ij})^2 \right) \right],$$

где

$$\alpha = (1.0, 1.2, 3.0, 3.2)^T$$

$$\mathbf{A} = \begin{pmatrix} 10 & 3 & 17 & 3.50 & 1.7 & 8 \\ 0.05 & 10 & 17 & 0.1 & 8 & 14 \\ 3 & 3.5 & 1.7 & 10 & 17 & 8 \\ 17 & 8 & 0.05 & 10 & 0.1 & 14 \end{pmatrix}$$

$$\mathbf{P} = 10^{-4} \begin{pmatrix} 1312 & 1696 & 5569 & 124 & 8283 & 5886 \\ 2329 & 4135 & 8307 & 3736 & 1004 & 9991 \\ 2348 & 1451 & 3522 & 2883 & 3047 & 6650 \\ 4047 & 8828 & 8732 & 5743 & 1091 & 381 \end{pmatrix}$$

Область определения:

$x_i \in [0, 1]$, i = 1, 2, 3, 4.

43. Функция Hartmann 6-Dimensional:

$$f(\mathbf{x}) = - \sum_{i=1}^{4} \alpha_i \exp\left(- \sum_{j=1}^{6} A_{ij} (x_j - P_{ij})^2 \right),$$

где

$$\alpha = (1.0, 1.2, 3.0, 3.2)^T$$

$$\mathbf{A} = \begin{pmatrix} 10 & 3 & 17 & 3.50 & 1.7 & 8 \\ 0.05 & 10 & 17 & 0.1 & 8 & 14 \\ 3 & 3.5 & 1.7 & 10 & 17 & 8 \\ 17 & 8 & 0.05 & 10 & 0.1 & 14 \end{pmatrix}$$

$$\mathbf{P} = 10^{-4} \begin{pmatrix} 1312 & 1696 & 5569 & 124 & 8283 & 5886 \\ 2329 & 4135 & 8307 & 3736 & 1004 & 9991 \\ 2348 & 1451 & 3522 & 2883 & 3047 & 6650 \\ 4047 & 8828 & 8732 & 5743 & 1091 & 381 \end{pmatrix}$$

Область определения:

$x_i \in (0, 1)$, $i = 1, \ldots, 6$.

Глобальный минимум:

$f(\mathbf{x}^*) = -3.32237, \mathbf{x}^* = (0.20169, 0.150011, 0.476874, 0.275332, 0.311652, 0.6573)$

Модифицированная форма функции:

$$f(\mathbf{x}) = -\frac{1}{1.94}\left[2.58 + \sum_{i=1}^{4} \alpha_i \exp\left(-\sum_{j=1}^{6} A_{ij}(x_j - P_{ij})^2\right)\right],$$

где

$$\alpha = (1.0, 1.2, 3.0, 3.2)^T$$

$$\mathbf{A} = \begin{pmatrix} 10 & 3 & 17 & 3.50 & 1.7 & 8 \\ 0.05 & 10 & 17 & 0.1 & 8 & 14 \\ 3 & 3.5 & 1.7 & 10 & 17 & 8 \\ 17 & 8 & 0.05 & 10 & 0.1 & 14 \end{pmatrix}$$

$$\mathbf{P} = 10^{-4} \begin{pmatrix} 1312 & 1696 & 5569 & 124 & 8283 & 5886 \\ 2329 & 4135 & 8307 & 3736 & 1004 & 9991 \\ 2348 & 1451 & 3522 & 2883 & 3047 & 6650 \\ 4047 & 8828 & 8732 & 5743 & 1091 & 381 \end{pmatrix}$$

44. Функция Shekel:

$$f(\mathbf{x}) = -\sum_{i=1}^{m}\left(\sum_{j=1}^{4}(x_j - C_{ji})^2 + \beta_i\right)^{-1},$$

где

$$m = 10$$

$$\beta = \frac{1}{10}(1, 2, 2, 4, 4, 6, 3, 7, 5, 5)^T$$

$$\mathbf{C} = \begin{pmatrix} 4.0 & 1.0 & 8.0 & 6.0 & 3.0 & 2.0 & 5.0 & 8.0 & 6.0 & 7.0 \\ 4.0 & 1.0 & 8.0 & 6.0 & 7.0 & 9.0 & 3.0 & 1.0 & 2.0 & 3.0 \\ 4.0 & 1.0 & 8.0 & 6.0 & 3.0 & 2.0 & 5.0 & 8.0 & 6.0 & 7.0 \\ 4.0 & 1.0 & 8.0 & 6.0 & 7.0 & 9.0 & 3.0 & 1.0 & 2.0 & 3.0 \end{pmatrix}$$

Область определения:

$x_i \in [0, 10]$, $i = 1, 2, 3, 4$.

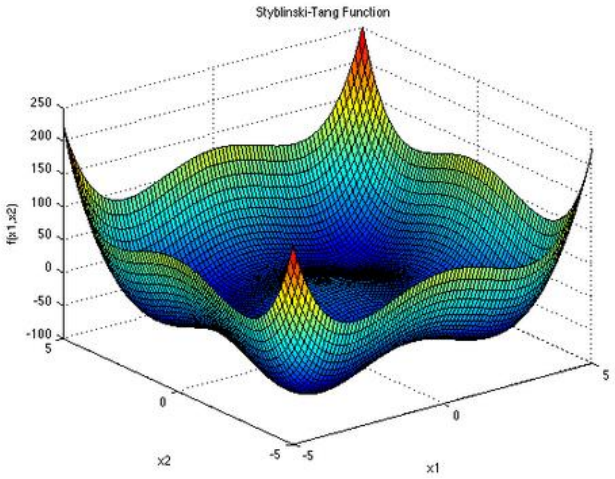
Глобальный минимум:

$m = 5$: $f(\mathbf{x}^*) = -10.1532$, $\mathbf{x}^* = (4, 4, 4, 4)$

$m = 7$: $f(\mathbf{x}^*) = -10.4029$, $\mathbf{x}^* = (4, 4, 4, 4)$

$m = 10$: $f(\mathbf{x}^*) = -10.5364$, $\mathbf{x}^* = (4, 4, 4, 4)$

45. Функция Styblinski-Tang:



$$f(\mathbf{x}) = \frac{1}{2}\sum_{i=1}^{d}(x_i^4 - 16x_i^2 + 5x_i)$$

Область определения:

$x_i \in [-5, 5]$, $i = 1, \ldots, d$.

Глобальный минимум:

$$f(\mathbf{x}^*) = -39.16599d, \quad \mathbf{x}^* = (-2.903534, \ldots, -2.903534)$$

46. Функция De Jong'a 1 (j1):

$$F(x_1, x_2) = \frac{100}{100\left(x_1^2 - x_2\right) + \left(1 - x_1\right)^2 + 1}$$

Область определения:

$$-1{,}28 \le x_{1,2} \le 1{,}28$$

$$F^* = F(1.00, 1.00) = 100.00$$

47. Функция De Jong'a 2 (j2):

$$F(x_1, x_2, \ldots x_5) = \sum_{i=1}^{5}\lceil x_i \rceil$$

Область определения:

$$-5{,}12 \le x_{1,2,3,4,5} \le 5{,}12$$

Глобальный минимум:

$F^* = 30$ на гиперплоскости $5{,}00 < x_{1,2,3,4,5} \le 5{,}12$

48. Функция De Jong'a 3 (j3):

$$F(x_1, x_2) = 0{,}002 + \sum_{j=1}^{25} \frac{1}{j + \sum_{i=1}^{2} (x_i - a_{ij})^6}$$

Область определения:

$$-65{,}536 \le x_{1,2,3,4,5} \le 65{,}536$$

Глобальный минимум:

$$F^* = F(-16, -32) = 1{,}002$$

49. Функция De Jong'a 4 (j4):

$$F(x_1, x_2) = 20 + x_1^2 + x_2^2 - 10\cos(2\pi x_1) - 10\cos(2\pi x_2)$$

Область определения:

$$-5{,}12 \le x_{1,2} \le 5{,}12$$

Глобальный минимум:

F^* = F *(4.52299, 4.52299)* = F *(-4.52299, 4.52299)* =
= F *(-4.52299, -4.52299)* = F *(4.52299, -4.52299) = 80.7065.*

50. Функция De Jong'a 5 (j5):

$$F(x_1, x_2) = \frac{1}{\frac{x_1^2 + x_2^2}{200} - \cos(x_1)\cos\left(\frac{x_2}{\sqrt{2}}\right) + 2}$$

Область определения:

$$-20.0 \le x_{1,2} \le 20.0$$

Глобальный минимум:

F^* = F *(0.00, 0.00) = 1.0.*

В качестве генетических операторов получения новых генотипов "потомков", используя генетическую информацию хромосомных наборов родителей мы применяли два типа кроссоверов - одно- и двухточечный. Вычислительные эксперименты показали, что даже для простых функций нельзя говорить о преимуществе того или иного оператора. Более того было показано, что использование механизма случайного выбора одно- или двух точечного кроссовера для каждой конкретной брачной пары подчас оказывается более эффективным, чем детерминированный подход к выбору кроссоверов, поскольку достаточно трудно априорно определить который из двух операторов более подходит для каждого конкретного ландшафта приспособленности. Из диаграмм,

представленных на рис. 4.12 видно, что нельзя однозначно отдавать предпочтение одно- или двухточечному кроссоверу. Одноточечный оказался более эффективным на тестовых функциях De Jong'a 2 и 5, на двумерной функции Griewank'a и на двумерной функции Растригина, однако для функции De Jong'a 3, функции Griewank'a и Растригина от 10 переменных можно говорить о преимуществе выбора двухточечного оператора. Использование же случайного выбора преследовало целью прежде всего сгладить различия этих двух подходов и улучшить показатели среднего ожидаемого результата. Для всех представленных тестовых функций так и произошло, - случайного выбор оказался эффективнее худшего. Кроме того, в ряде случаев (функции Griewank'a, 10-мерная функция Растригина) применение случайного механизма в выборе кроссовера дало лучшие результаты по сравнению с детерминированными подходами.

Повышение эффективности поиска при использовании случайного выбора операторов кроссовера повлияло на то, чтобы применить аналогичный подход при реализации процесса мутагинеза новых особей, однако в этом случае преимущество перед детерминированным подходом не так очевидно в силу традиционно малой вероятности мутации (в наших экспериментах вероятность мутации составляла 0.001 - 0.01).

Выбор родительской пары

Как уже отмечалось, мы решили не использовать вероятность кроссовера в качестве одного из параметров алгоритма, ограничивая число воспроизводимых потомков фиксированным числом брачных пар. В связи с этим встал вопрос о том, каким образом из всего многообразия всех возможных пар выбрать лишь несколько. Предлагалось несколько подходов, однако мы рассмотрим здесь несколько показавшихся нам наиболее интересными.

Первый подход самый простой - это случайный выбор родительской пары ("панмиксия"), когда обе особи, которые составят родительскую пару, случайным образом выбираются из всей популяции, причем любая особь может стать членом нескольких пар. Несмотря на простоту, такой подход универсален для решения различных классов задач. Однако он достаточно критичен к численности популяции, поскольку эффективность алгоритма, реализующего такой подход, снижается с ростом численности популяции.

Второй способ выбора особей в родительскую пару - так называемый селективный. Его суть состоит в том, что "родителями" могут стать только те особи, значение приспособленности которых не меньше среднего значения приспособленности по популяции, при равной вероятности таких кандидатов составить брачную пару. Такой подход обеспечивает более быструю сходимость алгоритма. Однако из-за быстрой сходимости селективный выбор родительской пары не подходит тогда, когда ставиться задача определения нескольких экстремумов, поскольку для таких задач алгоритм, как правило, быстро сходится к одному из решений. Кроме того, для некоторого класса задач со сложным ландшафтом приспособленности быстрая сходимость может превратиться в преждевременную сходимость к квазиоптимальному решению. Этот недостаток может быть отчасти компенсирован использованием подходящего механизма отбора (о чем будет сказано ниже), который бы "тормозил" слишком быструю сходимость алгоритма.

Другие два способа формирования родительской пары, на которые хотелось бы обратить внимание, это инбридинг и аутбридинг. Оба эти метода построены на формировании пары на основе близкого и дальнего "родства" соответственно. Под "родством" здесь понимается расстояние между членами популяции как в смысле геометрического расстояния особей в пространстве параметров (для фенотипов), так и в смысле хэмминингого расстояния между хромосомными наборами особей (для генотипов). В связи с этим будем различать генотипный и фенотипный (или географический) инбридинг и аутбридинг. Под инбридингом понимается такой метод, когда первый член пары выбирается случайно, а вторым с большей вероятностью будет максимально близкая к нему особь. Аутбридинг же, наоборот, формирует брачные пары из максимально далеких особей. Использование генетических инбридинга и аутбридинга оказалось более эффективным для всех тестовых функций при различных параметрах алгоритма. Наиболее полезно применение обоих представленных методов для многоэкстремальных задач. Однако два этих способа по-разному влияют на поведение генетического алгоритма. Так инбридинг можно охарактеризовать свойством концентрации поиска в локальных узлах, что фактически приводит к разбиению популяции на отдельные локальные группы вокруг подозрительных на экстремум участков ландшафта, напротив аутбридинг как раз направлен на

предупреждение сходимости алгоритма к уже найденным решениям, заставляя алгоритм просматривать новые, неисследованные области.

Механизм отбора

Обсуждение вопроса о влиянии метода создания родительских пар на поведение генетического алгоритма невозможно вести в отрыве от реализуемого механизма отбора при формировании нового поколения. Сразу же отметим, что метод пропорционального отбора, как и другие методы, основанные на включение особи в новую популяцию по вероятностному принципу, давали настолько плохие результаты, что здесь даже не рассматриваются. В своих экспериментах мы использовали другие механизмы отбора, из которых выделим два: элитный и отбор с вытеснением.

Идея элитного отбора, в общем, не нова, этот метод основан на построении новой популяции только из лучших особей репродукционной группы, объединяющей в себе родителей, их потомков и мутантов. Быстрая сходимость, обеспечиваемая элитным отбором, может быть, когда это необходимо, с успехом компенсирована подходящим методом выбора родительских пар, например аутбридингом (рис.4.9-4.11).

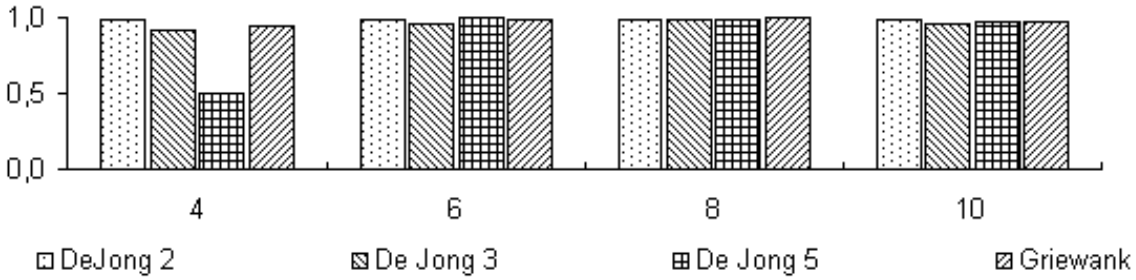


Рис.4.9. Отношение найденного максимального значения к реальному максимуму при различной длине кодировки гена для 4 тестовых функций

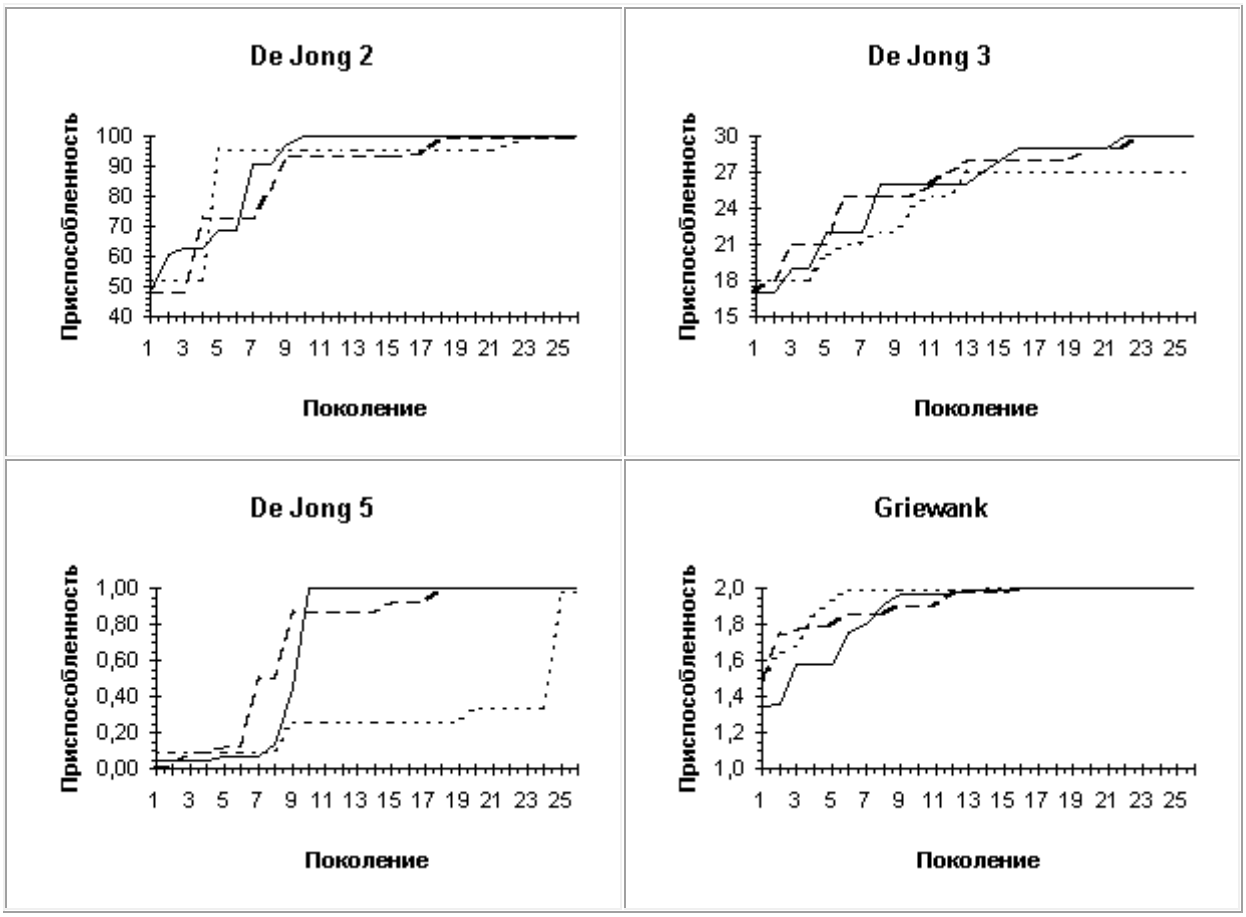


Рис.4.10. Графики изменения приспособленности лучшей особи в популяции при различной длине кодировки; непрерывная линия - L = 10, разрывная - L=6, пунктирная - L=4

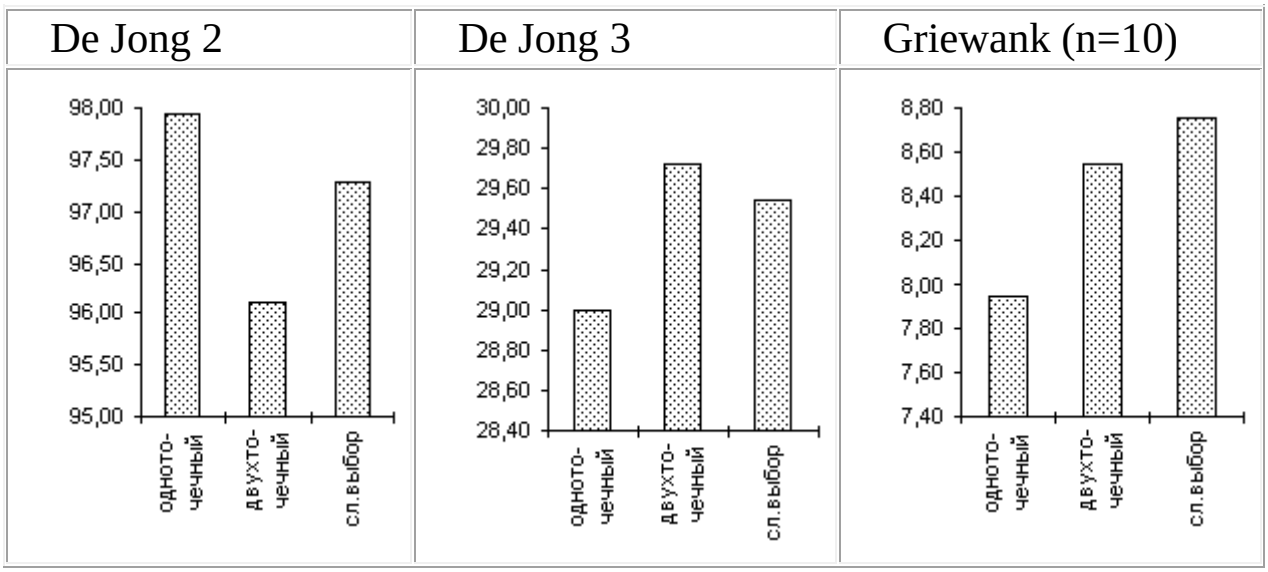


Рис.4.11. Эффективность ГА при использовании различных операторов кроссовера. На диаграммах представлены результаты, усредненные по 50 запускам (каждый запуск - около 5000 вычислений для 10-мерных функций или 1000 для остальных)

4.8. Исследование характеристик муравьиных алгоритмов

Исследования муравьиных алгоритмов начались в середине 90-х годов XX века, автором идеи является Марко Дориго из Университета Брюсселя, Бельгия [189]. В основе этих алгоритмов лежит моделирование поведения колонии муравьев. В ходе проделанной работы были реализованы и исследованы предложенные модификации муравьиного алгоритма (МА).

Пусть дан граф $G=(X,U)$, где $|X| = n$ – множество вершин (города), $|U| = m$ – множество ребер. Дана матрица чисел $D(i, j)$, где $1 \le i, j \le n$, представляющих собой стоимость ребра между вершинами x_i, x_j. Требуется найти перестановку φ из элементов множества X, такую, что значение целевая функция (ЦФ) равно:

$$Fitness(\varphi) = D(\varphi(1), \varphi(n)) + \sum_i \{D(\varphi(i), \varphi(i+1))\} \to \min .$$

Как было сказано выше, основная идея данного алгоритма является моделирование поведения муравьев, коллективной адаптации. Колония представляет собой систему с очень простыми правилами автономного поведения особей. Однако, несмотря на примитивность поведения каждого отдельного муравья, поведение всей колонии оказывается достаточно разумным [189]. Таким образом, основой поведения муравьиной колонии служит низкоуровневое взаимодействие, благодаря которому, в целом, колония представляет собой разумную многоагентную систему. Взаимодействие определяется через химическое вещество – феромон, откладываемый муравьями на пройденном пути. При выборе направления движения муравей исходит не только из желания пройти кратчайший путь, но и из опыта других муравьев, информацию о котором получают через уровень (количество) феромонов ребрах. С течением времени происходит процесс испарения феромонов, которое является отрицательной обратной связью.

Свойства муравья:

Каждый муравей обладает собственной «памятью», в котором будет храниться список городов $J_{i,k}$, которые необходимо посетить муравью k, который находится в городе i.

1. Муравьи обладают «зрением», обратно пропорциональным длине ребра:

$$\eta_{ij} = 1/D_{ij}.$$

2. Каждый муравей способен улавливать след феромона, который будет определять желание муравья пройти по данному ребру. Уровень феромона в момент времени t на ребре D_{ij} будет соответствовать $\tau_{ij}(t)$.

3. Вероятность перехода муравья из вершины i в вершину j будет определяться следующим соотношением:

$$\begin{cases} P_{ij,k}(t) = \dfrac{[\tau_{ij}(t)]^{\alpha} \cdot [\eta_{ij}(t)]^{\beta}}{\sum_{l \in J_{i,k}} [\tau_{il}(t)]^{\alpha} \cdot [\eta_{il}(t)]^{\beta}}, j \in J_{i,k}, \\ P_{ij,k}(t) = 0, j \notin J_{i,k} \end{cases} \quad (4.21)$$

где α, β – эмпирические коэффициенты [49]. Нетрудно заметить, что данное выражение имеет эффект «колеса рулетки». Количество откладываемого феромона определяется выражением:

$$\Delta\tau_{ij,k}(t) = \begin{cases} \dfrac{Q}{L_k(t)}, (i,j) \in T_k(t) \\ 0, (i,j) \notin T_k(t) \end{cases}, \quad (4.22)$$

где Q – параметр, имеющий значение порядка длины оптимального пути, $L_k(t)$ – длина маршрута $T_k(t)$. Испарение феромона определяется следующим выражением:

$$\tau_{ij}(t+1) = (1-p) \cdot \tau_{ij}(t) + \sum_{k=1}^{m} \Delta\tau_{ij,k}(t), \quad (4.23)$$

где m – количество муравьев, p – коэффициент испарения $(0 \le p \le 1)$.

4.8.1. Параллельный муравьиный алгоритм оптимизации

Муравьиный алгоритм достаточно просто поддаётся разбиению на независимо выполняемые фрагменты. Причём несколькими различными способами, выбор которых зависит от поставленной цели, например: сокращение времени выполнения, повышение точности решений, более адекватное моделирование поведения муравьёв и т.д.

Чаще всего создание параллельных алгоритмов преследует цель сокращения времени решения задач большой размерности. Например, в работе [189] автором предлагается такая схема параллельного алгоритма, когда наиболее ресурсоёмкий этап построения и улучшения маршрутов для каждого муравья производится абсолютно независимо в отдельных потоках. Здесь мы более подробно рассмотрим несколько иной подход к реализации параллельного алгоритма оптимизации муравьиной колонии, направленный на

достижение максимально приемлемого результата за однократное выполнение, на примере задачи коммивояжера. Суть алгоритма сводится к тому, что создаваемая колония муравьёв разбивается на n групп по m муравьёв в каждой. Группы функционируют в отдельных процессах (выполняют свои задачи на отдельном процессоре). Условия задачи идентичны для всех групп, т.е. в каждой отрабатывается алгоритм решения задачи коммивояжера в его обычном виде, и в результате формируется лучший путь в пределах g-ой группы

$$d_g(C) = \sum_{(i,j)\in A} d(i,j) \to \min,\ g = \overline{1,n}. \qquad (4.24)$$

Отличием данного алгоритма является введение дополнительной процедуры, осуществляющей на протяжении заданного количества попыток поиска оптимального маршрута периодическое глобальное обновление феромонной матрицы с выделением лучшего пути, найденного среди всех n групп. Данное обновление учитывается при очередной внутригрупповой итерации, и рёбра графа получают усиление феромоном в соответствии с ним. По завершении всех попыток поиска выбирается наилучший, найденный в конечном итоге, маршрут среди n групп, который и выдаётся в качестве окончательного решения:

$$d(C) = \min_{g=1,n}\{d_g(C)\}.$$

Алгоритм, таким образом, обеспечивает не только внутригрупповую, но межгрупповую глобальную коммуникацию, что положительно влияет на повышение точности решения. Дополнительно для ускорения работы групп в алгоритме применяется стратегия элитных муравьёв.

На рисунках 4.12 и 4.13 представлены результаты эксперимента, проведённого на вычислительном кластере Центра разработки программных продуктов и аппаратно-программных комплексов при Ташкентском Университете информационных технологий.

Вычислительный эксперимент проводился при следующих заданных условиях задачи: $N = 100$, $m = 100$, $\alpha = 1.0$, $\beta = 2.0$, $\rho = 0.20$, $Q = 80$, число попыток поиска – 35. Глобальное обновление лучшего маршрута среди групп – каждые 5 итераций.

Как видно из рисунков 4.12 и 4.13 с ростом количества задействованных процессоров при однократном выполнении алгоритма длина искомого кратчайшего маршрута сокращается, то

есть определяется более оптимальное решение задачи. При этом время работы программы при заданных параметрах увеличивается незначительно, и связанно с ростом числа межпроцессорных обменов.

Несмотря на такую положительную динамику, нецелесообразное увеличение количества задействованных процессоров не желательно. Требуется соблюдать баланс между условиями/параметрами задачи и числом процессоров, так как с определённого момента алгоритм перестанет выдавать улучшенные решения вне зависимости от их количества. Кроме того, на качество получаемого решения заметно влияет аппаратно-программная платформа вычислительного полигона, на котором осуществляется компиляция и выполнение программы. В том числе и качество используемого генератора псевдослучайных чисел.

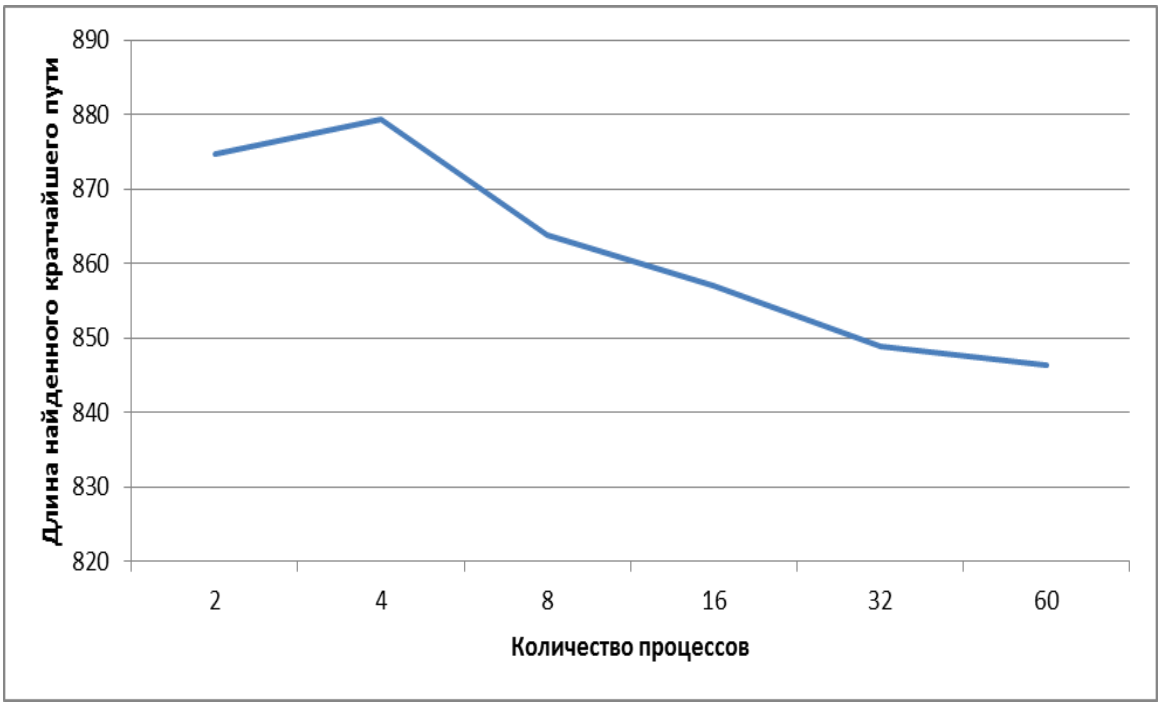


Рис. 4.12. Динамика изменения качества решения в зависимости от увеличения количества используемых процессоров

Рис. 4.13. Изменение времени работы алгоритма с увеличением количества процессоров

Основная идея, лежащая в основе алгоритмов муравьиной колонии, заключается в использовании механизмов положительной и отрицательной обратных связей, которые помогают найти наилучшее приближённое значение общего (глобального) решения, не зацикливаясь на локальных (субоптимальных) решениях исследуемых задач оптимизации. Важной особенностью структуры муравьиных алгоритмов является возможность распараллеливания их процедур. Проведённые вычислительные эксперименты с использованием предложенного подхода к распараллеливанию муравьиного алгоритма показали, что они обеспечивают существенное сокращение времени вычислений и повышение точности общих (глобальных) решений.

Глава 5. АЛГОРИТМЫ КВАНТОВОЙ ОПТИМИЗАЦИИ

5.1.Решение задачи нелинейной оптимизации на основе алгоритма Гровера

Алгоритм Гровера - это квантовый алгоритм, разработанный для поиска элемента в неотсортированном списке базовой структуры данных. Он позволяет найти элемент с определенным свойством в списке функций, используя квантовые вычисления. С другой стороны, задачи нелинейной оптимизации направлены на нахождение минимума или максимума целевой функции, которая может быть нелинейной и может иметь различные ограничения. Для решения таких задач используются различные алгоритмы оптимизации, такие как генетические алгоритмы, методы градиентного спуска, алгоритмы на основе симуляции отжига и др. В контексте квантовых вычислений, существует идея использовать квантовые методы для оптимизации, включая алгоритмы квантовой оптимизации, которые включают в себя квантовые вариационные алгоритмы и квантовые методы оптимизации без градиентов. Эти квантовые методы оптимизации могут использоваться для решения задач нелинейной оптимизации, используя преимущества квантовых вычислений в вычислительной мощности.

Оптимизация играет важную роль в многих областях, включая машинное обучение и промышленность. Поиск оптимальных решений часто является вычислительно сложной задачей, и квантовые вычисления предоставляют потенциал для значительного ускорения этого процесса. Однако, есть некоторые ограничения и проблемы с применением квантовых алгоритмов к оптимизационным задачам. В настоящее время доступные квантовые компьютеры все еще ограничены по числу кубитов и степени свободы. Это ограничивает их применимость к решению сложных оптимизационных задач с большим числом переменных. Квантовые вычисления подвержены ошибкам и потере квантовой информации из-за внешних факторов [201-203]. Для эффективной оптимизации требуется высокая точность, и управление ошибками в квантовых системах является активной областью исследований. Разработка квантовых алгоритмов для конкретных оптимизационных задач может быть сложной задачей, требующей глубоких знаний в области квантовой информатики. Это ограничивает доступность и применимость квантовых алгоритмов для широкого круга

специалистов. На данный момент инфраструктура для квантовых вычислений, такие как квантовые компьютеры, квантовые языки программирования и т. д. все еще находится на стадии развития. Это ограничивает доступность квантовых ресурсов для многих исследователей и компаний. Несмотря на эти ограничения, квантовые вычисления предоставляют потенциал для решения оптимизационных задач более эффективно, чем классические методы. С развитием квантовых технологий и алгоритмов можно ожидать, что квантовые алгоритмы будут играть все более важную роль в оптимизации и решении сложных промышленных проблем [204].

Для использования квантовых алгоритмов необходимо сначала сформулировать задачу в квантовой форме, что включает в себя кодирование данных и операций над ними на квантовых битах (кубитах). Процесс формулирования задачи для квантового алгоритма может включать следующие шаги [205]:

Определение, может ли задача быть сформулирована как оптимизационная задача, поиск максимума или минимума функции, или другая задача, которую можно решить с помощью квантового алгоритма.

Определение, какие данные необходимо закодировать в квантовые биты (кубиты). Кодирование данных на квантовом уровне может потребовать разработки специальных квантовых схем, чтобы представить информацию в виде состояний кубитов.

Разработка квантовой программы или квантовой схемы, которая выполняет операции над закодированными данными. Эти операции могут включать в себя квантовые вентили, квантовые вращения и другие квантовые преобразования. В зависимости от алгоритма вам может потребоваться использовать процедуры амплификации амплитуд, чтобы улучшить вероятность правильного ответа, и, наконец, произвести измерение кубитов, чтобы получить результаты задачи.

Измерение результата квантового алгоритма должны быть адекватно проанализированы и интерпретированы.

Эти шаги представляют общий процесс формулирования и решения задач с использованием квантовых алгоритмов. Важно помнить, что не все задачи подходят для квантовых алгоритмов, и выбор алгоритма и архитектуры зависит от конкретной задачи и возможностей доступных квантовых устройств [206].

Оракулы представляют собой ключевой элемент, который позволяет кодировать задачу оптимизации или другую задачу для квантового решения. Сначала необходимо явно определить задачу, которую нужно решить с использованием квантовых алгоритмов оптимизации. Например, это может быть задача поиска оптимального решения для некоторой функции. Затем вам нужно закодировать данные задачи в квантовые биты (кубиты). В случае оптимизации это может включать значения параметров, которые следует оптимизировать. Можно представить каждый параметр как набор кубитов, где каждый кубит кодирует определенное значение параметра. Далее, для решения задачи оптимизации, надо создать квантовый оракул. Оракул представляет собой квантовую схему, которая может выполнить операции над закодированными данными. Он может включать в себя квантовые вентили и операторы, которые изменяют состояния кубитов в зависимости от задачи. Например, он может инвертировать состояния, соответствующие лучшим решениям задачи, чтобы увеличить их вероятность. После применения оракула можно использовать амплификацию амплитуд, чтобы увеличить вероятность правильного ответа. Это может потребовать несколько итераций, чтобы улучшить вероятность. Наконец, измеряется состояние кубитов, чтобы получить ответ на задачу оптимизации. Измерения дают вероятностное распределение результатов, и на основе этого распределения оценивается оптимальное решение [207].

Квантовые алгоритмы оптимизации предоставляют возможность искать лучшие решения в более эффективном режиме, чем классические методы. Кодирование проблемы оптимизации в кубиты и использование квантовых операций для изменения состояний кубитов позволяют оценивать качество решений. Задачу оптимизации можно представить в виде функции, которая принимает набор параметров и возвращает "функцию стоимости", которую нужно минимизировать или максимизировать. Значения параметров могут быть закодированы в кубиты, где каждый кубит представляет значение параметра. Квантовые алгоритмы оптимизации, такие как алгоритм Гровера и квантовые вариационные алгоритмы, предоставляют новые инструменты для решения сложных задач оптимизации. Эти алгоритмы позволяют искать лучшие решения в пространствах параметров, которые классическим методам было бы сложно обойти [201,208].

Актуальность оптимизации на основе квантового алгоритма Гровера продолжает расти из-за нескольких ключевых факторов. Алгоритм Гровера предоставляет квантовые преимущества при решении задач оптимизации по сравнению с классическими методами. Эти преимущества проявляются в ускорении поиска оптимальных решений, особенно при работе с большими объемами данных и сложными функциями. С развитием квантовых технологий становится доступным более мощное и масштабируемое аппаратное обеспечение для реализации алгоритма Гровера. Это позволяет исследователям и инженерам применять этот алгоритм к различным задачам оптимизации. Алгоритм Гровера изначально разрабатывался для задачи поиска, но его применение было расширено на задачи оптимизации. Сейчас он применяется в таких областях, как машинное обучение, криптография, оптимизация сложных систем и финансы. В некоторых областях, таких как оптимизация параметров в глубоком обучении или решение сложных задач дискретной оптимизации, классические методы могут сталкиваться с ограничениями. Алгоритм Гровера может помочь находить более эффективные решения в таких задачах. Оптимизация на основе алгоритма Гровера может интегрироваться в более широкий контекст квантовых вычислений, такой как использование квантовых аппаратов, разработка квантовых алгоритмов и платформ для исследований в области оптимизации. В целом, актуальность оптимизации на основе алгоритма Гровера обусловлена его потенциалом ускорения поиска оптимальных решений в различных областях и его интеграцией в развивающуюся экосистему квантовых вычислений [201,209].

Основная цель оптимизации на основе квантового алгоритма Гровера заключается в поиске оптимального значения функции в заданном пространстве поиска. Алгоритм Гровера может быть использован для поиска глобального минимума или максимума в заданной функции. Это позволяет решать задачи оптимизации, такие как оптимизация параметров машинного обучения, поиск оптимальных конфигураций систем или оптимизация функций стоимости в финансовых приложениях и может помочь ускорить процесс оптимизации, сокращая количество итераций, необходимых для нахождения оптимального решения. Это особенно полезно, когда вычисления на классическом компьютере могут быть вычислительно затратными. Алгоритм Гровера может быть применен к сложным задачам оптимизации с большим числом переменных и сложными

функциями. Он может помочь находить оптимальные решения в таких задачах, которые были бы трудноподдающимися классическим методам оптимизации. Оптимизация на основе алгоритма Гровера может быть интегрирована в квантовые вычисления, что делает ее частью более широкой экосистемы квантовых приложений. В целом, цель оптимизации на основе алгоритма Гровера заключается в улучшении производительности поиска оптимальных решений в различных областях, используя квантовые преимущества этого алгоритма [201,210].

Основные задачи оптимизации, которые можно решать на основе квантового алгоритма Гровера, включают следующие [201,211]:

Поиск в неупорядоченном списке (Unstructured Search): Исходной алгоритм Гровера был разработан для решения этой задачи. Алгоритм может быстро найти нужный элемент в списке, даже если он находится среди большого числа других элементов.

Решение задачи SAT (Satisfiability Problem): SAT - это задача определения, существует ли такая комбинация значений булевых переменных, которая удовлетворяет заданной булевой формуле. Алгоритм Гровера может использоваться для поиска такой комбинации, если она существует.

Оптимизация параметров машинного обучения: Алгоритм Гровера может использоваться для оптимизации параметров моделей машинного обучения, таких как нейронные сети. Он может помочь найти наилучшие значения параметров, минимизирующие функцию потерь.

Решение задачи коммивояжера (Traveling Salesman Problem, TSP): TSP - это задача нахождения кратчайшего пути, который проходит через все заданные города и возвращается в начальный город. Алгоритм Гровера может использоваться для поиска оптимального маршрута.

Оптимизация портфеля инвестиций: В финансовой области алгоритм Гровера может использоваться для оптимизации выбора инвестиционного портфеля, который максимизирует ожидаемую прибыль при заданных ограничениях.

Дискретная оптимизация: Алгоритм Гровера может применяться к различным задачам дискретной оптимизации, таким как задачи на графах, планирование, раскрой и другие.

Криптография: В криптографии алгоритм Гровера используется для атаки на криптографические хеш-функции и расшифровки некоторых шифров.

Решение NP-трудных задач: Многие NP-трудные задачи, для которых известны алгоритмы с экспоненциальной сложностью, могут быть решены более эффективно с использованием алгоритма Гровера.

Основное преимущество алгоритма Гровера заключается в его способности квантово ускорять поиск оптимальных решений в таких задачах, что делает его ценным инструментом в различных областях, где требуется оптимизация.

На данный момент существует несколько квантовых алгоритмов и подходов для решения различных задач оптимизации. Эти алгоритмы стремятся использовать преимущества квантовых вычислений для решения задач оптимизации более эффективно, чем классические алгоритмы. Некоторые из наиболее известных квантовых алгоритмов оптимизации включают [212-214]:

1.Вариационный Квантовый Решатель (Variational Quantum Solver, VQS): Этот алгоритм использует вариационные цепи для настройки параметров квантовой схемы, позволяя приближаться к оптимальному решению. Он широко применяется для решения задач оптимизации, таких как задачи портфельного управления и оптимизация химических молекул.

Общий алгоритм для задач оптимизации:

1.Инициализация квантового состояния: Создайте начальное квантовое состояние, которое представляет потенциальное решение задачи. Это может быть суперпозиция различных состояний кубитов.

2.Квантовый оракул: Создайте квантовый оракул, который выполняет операцию, соответствующую функции, которую вы оптимизируете. Этот оракул оценивает функцию на текущем состоянии и возвращает информацию о значении функции.

3.Квантовая адаптация: Используйте операции на кубитах для адаптации квантового состояния на основе информации, предоставленной оракулом. Это может включать в себя операторы вращения (гейты) и амплификацию состояний.

4. Итерации: Повторяйте шаги 2 и 3 некоторое количество раз, чтобы квантовое состояние сходилось к оптимуму.

5. Измерение и классическая обработка: После завершения квантовых операций, измерьте состояние кубитов и получите

результаты. Затем используйте классический алгоритм для анализа результатов и определения оптимального решения.

Повторение: Повторяйте алгоритм с разными начальными состояниями и параметрами, чтобы найти наилучшее решение.

2.Алгоритм Амплификации Амплитуд (Amplitude Amplification Algorithm): Этот алгоритм основан на амплификации амплитуды правильных решений и применяется для поиска максимумов и минимумов функций.

Алгоритм Амплификации Амплитуд (Amplitude Amplification Algorithm) - это квантовый алгоритм, разработанный для увеличения амплитуды правильных ответов в квантовых состояниях, что может быть полезным для решения задач оптимизации, поиска, и других задач.

Основной идеей алгоритма является использование повторяющихся операций, включая оператор инверсии среднего значения и оператор оракула, чтобы постепенно увеличивать вероятность обнаружения правильных решений. Вот общий алгоритм:

Инициализация состояния: Начните с исходного квантового состояния, которое может быть суперпозицией различных состояний.

Итерации оператора Гровера: Повторяйте следующие два шага некоторое количество раз:

a. Примените оператор оракула: Это квантовая операция, которая отмечает правильные ответы и изменяет амплитуды соответствующих состояний.

b. Примените оператор инверсии среднего значения: Этот оператор инвертирует амплитуды относительно среднего значения всех состояний. Он способствует увеличению амплитуд правильных ответов.

Измерение состояния: После завершения итераций оператора Гровера измерьте состояние. Вероятность обнаружения правильных ответов будет значительно выше, чем в случае, если бы вы использовали только случайный поиск.

3. Variational Quantum Eigensolver (VQE) - вариант вариационного квантового алгоритма, который используется для приближенного решения квантовых задач, таких как вычисление энергии молекулы.

Вариационные квантовые алгоритмы, включая VQE, строят параметризованный квантовый циркуит, который зависит от набора

параметров. Затем они оптимизируют эти параметры классическими методами так, чтобы минимизировать ожидаемое значение некоторого оператора (например, оператора Гамильтона) относительно квантового состояния. В случае VQE это означает приближенное вычисление энергии основного состояния квантовой системы.

Вот общий шаги VQE:

Подготовка квантового состояния: Инициализация квантового состояния, например, в виде суперпозиции базисных состояний.

Построение параметризованного циркуита: Создание квантовой схемы, которая зависит от набора параметров (обычно называемых "вариационными параметрами").

Оптимизация параметров: Используйте классический оптимизационный алгоритм (например, градиентный спуск) для нахождения наилучших значений вариационных параметров, минимизирующих ожидаемое значение оператора Гамильтона.

Вычисление результата: После оптимизации параметров измерьте ожидаемое значение оператора Гамильтона на полученном квантовом состоянии. Это будет приближенной энергией основного состояния системы.

VQE может быть применен к различным задачам оптимизации, таким как оптимизация химических молекул, поиск параметров в машинном обучении, и многим другим. Этот метод остается активной областью исследований в квантовом вычислении и имеет потенциал для решения ряда сложных оптимизационных задач.

4.Квантовый Генетический Алгоритм (Quantum Genetic Algorithm, QGA): Этот метод объединяет идеи генетических алгоритмов с квантовыми преобразованиями для решения задач оптимизации.

Квантовый Генетический Алгоритм представляет собой алгоритм, который комбинирует концепции генетических алгоритмов с квантовыми преобразованиями для решения задач оптимизации. Он представляет собой интересное сочетание классических методов оптимизации и квантового вычисления. Вот общий алгоритм QGA:

1. Инициализация популяции: Начните с создания случайной популяции кандидатов на решение задачи оптимизации.

2.Применение квантовых операторов: Каждый кандидат представляется в виде квантового состояния. Затем применяются

квантовые преобразования, такие как преобразование Адамара или другие квантовые вращения, для обработки каждого состояния.

3.Вычисление функции пригодности: Для каждого кандидата в популяции вычисляется значение функции пригодности (целевой функции), которую необходимо оптимизировать.

4.Селекция: Выбираются лучшие кандидаты на основе значений функции пригодности. Эти кандидаты будут использоваться для создания новой популяции.

5.Генетические операторы: Применяются генетические операторы, такие как скрещивание и мутации, к выбранным кандидатам, чтобы создать новую популяцию.

6.Повторение: Шаги с 2 по 5 повторяются несколько раз до достижения условия остановки.

7. Результат: По завершении алгоритма выбирается лучшее найденное решение.

Основной особенностью QGA является то, что кандидаты на решение представлены в виде квантовых состояний, и квантовые операторы используются для их обработки. Это позволяет алгоритму использовать преимущества квантового параллелизма и интерференции для более эффективного поиска оптимальных решений.

5.Квантовый Вариационный Метод (Quantum Variational Method, QVM): Этот метод в квантовом вычислении базируется на принципах вариационного метода и квантовых вычислений. Основные идеи и шаги QVM включают в себя:

Подготовка параметрического квантового состояния: Начните с инициализации квантового состояния, которое зависит от параметров (вариационных параметров). Эти параметры можно рассматривать как переменные, которые нужно оптимизировать.

Построение параметрического квантового циркуита: Создайте квантовый циркуит, который зависит от параметров, включая операторы вращения и другие квантовые операции. Этот циркуит будет представлять кандидата на решение задачи.

Оптимизация параметров: Используйте классический оптимизационный алгоритм, такой как градиентный спуск или другие методы оптимизации, для нахождения наилучших значений параметров, минимизирующих целевую функцию (функцию пригодности).

Измерение результатов: После оптимизации параметров выполните измерения на квантовом состоянии, чтобы получить результат задачи оптимизации или анализа.

QVM может быть применен к различным задачам, включая оптимизацию химических молекул, решение задач машинного обучения, анализ квантовых систем и многие другие. Он предоставляет способ использования квантовых преимуществ для решения сложных задач оптимизации, путем настройки параметров квантового циркуита.

6.Гауссовский Процесс Квантового Моделирования (Quantum Gaussian Process Modeling, QGPM): Этот метод использует гауссовские процессы и квантовые компьютеры для решения оптимизационных задач.Основные идеи и шаги QGPM включают в себя:

Создание модели Гауссовского процесса: Начните с создания гауссовской модели, которая описывает функцию, которую вы хотите аппроксимировать или оптимизировать. Эта модель может использоваться для предсказания значений функции в различных точках.

Квантовое улучшение модели: Затем используйте квантовые вычисления, такие как квантовые симуляторы или будущие квантовые компьютеры, для улучшения модели GP. Это может включать в себя обработку большего объема данных или улучшение точности аппроксимации.

Оптимизация модели: Применяйте методы оптимизации для настройки параметров GP модели, чтобы минимизировать ошибку аппроксимации или достичь других целей задачи оптимизации.

Использование модели: После настройки модели GP вы можете использовать ее для предсказания значений функции в новых точках или для решения других задач, таких как оптимизация.

Использование квантовых вычислений для улучшения GP моделей может быть особенно полезным в случаях, когда обработка больших объемов данных или анализ сложных функций требует больших вычислительных ресурсов. Квантовые методы могут ускорить этот процесс и повысить точность аппроксимации.

Каждый из этих методов имеет свои собственные преимущества и ограничения и может быть наилучшим выбором в зависимости от конкретной задачи оптимизации и доступных ресурсов квантового

компьютера. Кроме того, активно идет исследование новых квантовых алгоритмов оптимизации, и область развивается быстро.

Алгоритм Гровера изначально был разработан для поиска элемента в неотсортированном списке базовых данных с использованием квантовых вычислений. Однако его можно адаптировать и применить для задач оптимизации. Вот общий алгоритм Гровера для решения задачи оптимизации [215-216]:

1.Инициализация состояния: Создайте квантовое состояние, которое представляет набор параметров, подлежащих оптимизации. Это может быть выполнено, например, с использованием суперпозиции кубитов.

«Стартовый» блок устанавливает исходное значение **«x».** Например, если речь идёт о поиске решения в виде целого числа в диапазоне от **«0»** до **«N»**, тогда в качестве исходного значения **«x»** выбирают обычно наименьшее – **«0»**. Так же устанавливается в **«0»** служебная ячейка **«f».**

Мы работаем с кубитовым регистром. Размер регистра – количество кубитов – выбирается таким образом, чтобы гарантированно перекрыть весь диапазон поиска. Плюс добавляется один «флаговый» кубит. Ту часть регистра, где храняться проверяемые данные, обозначим как **«X»** и назовём это «подрегистр данных». Отдельные кубиты подрегистра данных обозначим как **«x1, x2 ... xk»**. Здесь **k** – это количество кубитов в регистре. Подразумеваем, что $\mathbf{x_k}$ – это старший разряд числа, $\mathbf{x_1}$ – младший. Квантовое состояние подрегистра данных будем записывать вот в таком формате: $\mathbf{|x_k, \ldots x_1\rangle}$**.** Так, как принято для чисел: старший разряд слева, младший - справа. Или будем писать сокращённо **|x⟩.** А «флаговый» кубит обозначим как **«f»**. Формируется в кубитах **«x1, x2 ... xk»** равновесную суперпозицию всех значений из области поиска [217].

Все кубиты подрегистра **«x»** приводим в состояние **|0⟩**. Технически операция «обнуления» осуществляется путём измерения кубита. Предполагается, что до начала работы все кубиты находятся в произвольном, как правило, неизвестном нам состоянии. Значит, в результате измерения мы можем получить как **⟨0⟩**, так и **⟨1⟩**. Если получен результат **⟨0⟩**, прекрасно, значит, кубит гарантированно находится в состоянии **|0⟩**, .оставляем его в покое до следующего шага. Если **⟨1⟩** – применяем к такому кубиту унитарную операцию **[X]**, она же - квантовый гейт **«NOT»**.

Каждый кубит подрегистра x находится теперь в чистом состоянии **|0⟩.** Значит, системное состояние подрегистра x мы можем записать как произведение состояний отдельных кубитов или как **k**-кубитное квантовое состояние [201,218]:

$$|\mathbf{x}\rangle = |\mathbf{0}_{x_k}\rangle * |\mathbf{0}_{x_{k-1}}\rangle * \dots * |\mathbf{0}_{x_2}\rangle * |\mathbf{0}_{x_1}\rangle = |\mathbf{0}_{x_k}\mathbf{0}_{x_{k-1}} \dots \mathbf{0}_{x_2}\mathbf{0}_{x_1}\rangle .$$

Флаговый кубит «f» устанавливается в состояние: Сначала переводим кубит «f» в состояние |1⟩, затем применяем к нему однокубитную операцию Адамара [H],

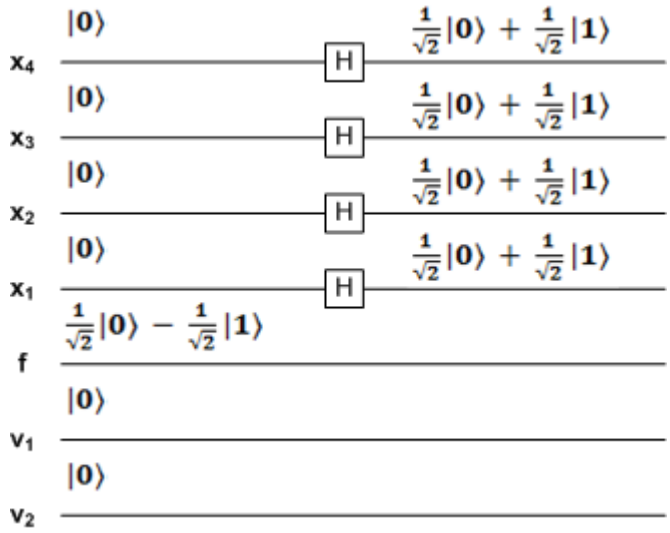
$$|\mathbf{f}\rangle = \frac{1}{\sqrt{2}}|\mathbf{0}\rangle - \frac{1}{\sqrt{2}}|\mathbf{1}\rangle .$$

В общем виде, для k-кубитного регистра данных, после инициализации состояние будет выглядеть так:

$$|\mathbf{x}\rangle = \frac{1}{(\sqrt{2})^k}\sum_{i=0}^{N-1}|\mathbf{p}_i\rangle .$$

Здесь **N** - это общее количество базисных состояний (представляемых чисел) в суперпозиции, определяемое как **2** в степени **k**.

Под **|pi⟩** в формулах подразумевается базисное **k**-кубитное состояние, соответсвующее двоичному представлению числа "**i**". И нарисуем вычислительную схему.



Пока кубиты не запутаны, мы вправе это делать, в смысле, рассматривать каждый кубит отдельно.

2. Применение оператора оракула: Создайте квантовый оракул, который отображает текущее состояние параметров на значение оптимизируемой функции. Этот оракул можно реализовать как оператор, который инвертирует амплитуду состояний, соответствующих лучшим параметрам (т.е., с наименьшим значением функции) [201,219,220].

Шаг 2 проверяет соответствие входного значения заданному критерию. На входе шага 2 присутствуют суперпозиция всех **N** проверяемых значений. На выход - суперпозиция всех **N** результатов проверки. В силу такого квантового параллелизма оракул проверяет все значения за один цикл работы.

Пусть, например, флаговый кубит находится в состоянии:

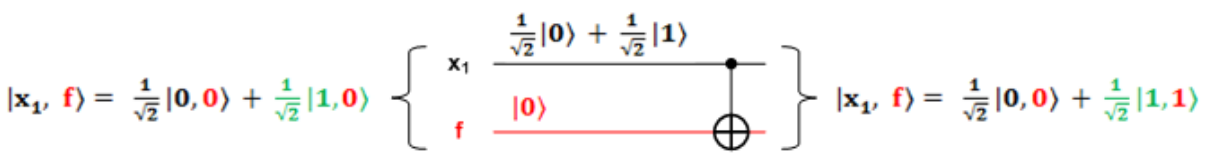
$$|f\rangle = |0\rangle.$$

В этом случае квантовое состояние системы кубитов $|x,f\rangle$ "на входе" блока 2 несёт в себе **N** базисных состояний - альтернатив следующего вида [221]:

$$|x_k, x_{k-1}, \ldots x_1, 0_f\rangle.$$

Процедура проверки строится таким образом, что состояние кубита «**f**» в "правильных" альтернативах меняется на противоположное. В нашем случае $|0\rangle$ меняется на $|1\rangle$. В прочих "неправильных" альтернативах, флаговый кубит сохраняет состояние $|0\rangle$.

Если же изначально флаговый кубит находится в состоянии $|1\rangle$, тогда в "правильных" альтернативах его состояние в результате работы оракула меняется на $|0\rangle$. Простейшим примером оракула является гейт «**CNOT**». Если рассматривать кубит на контролирующем входе как кубит данных, а «рабочий» кубит - как флаговый, тогда можно считать, что гейт «CNOT» проверяет кубит данных на предмет его равенства единице. У единственного кубита данных всего две альтернативы. Так вот, для той альтернативы, где кубит данных находится в состоянии $|1\rangle$, состояние флагового кубита меняется. Покажем это на рисунке для случая когда флаг инициализирован в $|0\rangle$ **[222]**:



"Правильная" альтернатива на этом рисунке и дальше выделена

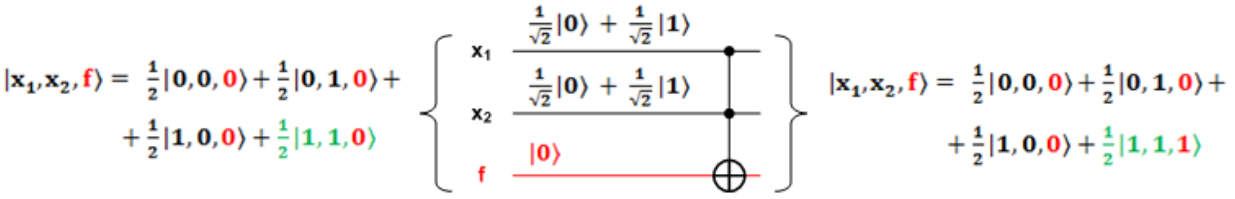
зелёным цветом. Дополнительно красным выделено всё то, что относится к кубиту **«f»**.

Заметим по ходу дела, что сепарабельное квантовое состояние (его формула на рисунке слева) в результате работы оракула меняется на запутанное состояние (формула справа). Математически сепарабельное состояние квантовой системы характеризуется тем, что его формула в виде суммы может быть корректно преобразована в произведение двух состояний отдельных подсистем. В частности, для формулы слева:

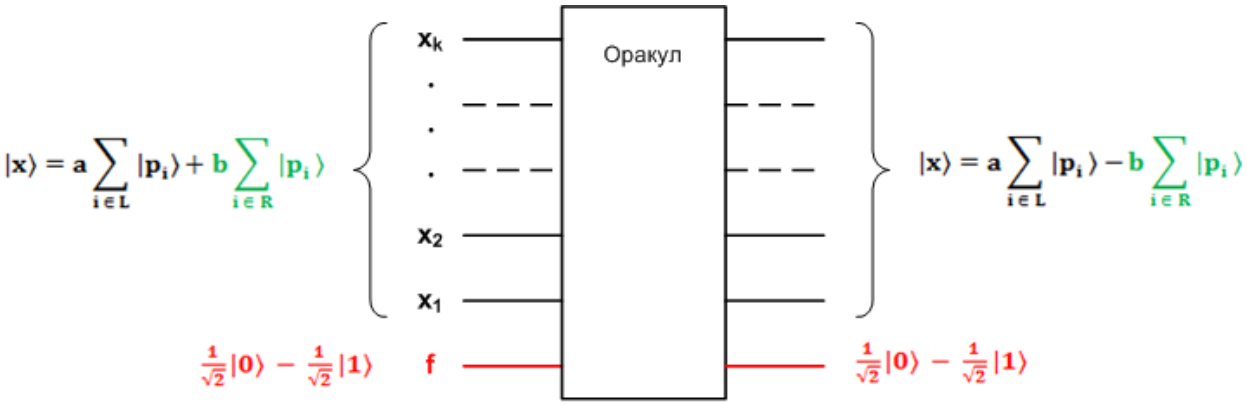
$$\frac{1}{\sqrt{2}}|0,0\rangle + \frac{1}{\sqrt{2}}|1,0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) * |0\rangle$$

.

Аналогично «маленьким оракулом» можно считать трёхкубитный гейт **«CCNOT»**. Он «переворачивает» флаг только для одной из четырёх альтернатив. А именно, для той, где оба контролирующих кубита равны единице.



Обобщённо, для любой задачи, оракул можно представить в виде вот такого "чёрного ящика" [1,23]:



В формулах на рисунке приняты следующие обозначения:

i ∈ L- суммирование по всем i, принадлежащим к множеству "неправильных" значений;

i ∈ R - суммирование по всем i, принадлежащим к множеству "правильных" значений.

a - амплитуды вероятности "неправильных" альтернатив;

b - амплитуды вероятности "правильных" альтернатив.

С каждым проходом алгоритма амплитуды **"a"** будут уменьшаться, а амплитуды **"b"** - увеличиваться. Почему - про это мы узнаем в следующем посте.

С каждым проходом алгоритма амплитуды **"a"** будут уменьшаться, а амплитуды **"b"** - увеличиваться.

3. Амплификация амплитуд: Применение оператора инверсии среднего значения: Этот оператор увеличивает амплитуду состояний, соответствующих лучшим параметрам, и уменьшает амплитуду остальных состояний. Это помогает сосредоточиться на оптимальных значениях [224,225].

После оракула амплитуды вероятности альтернатив, содержащих правильное значение, отрицательны. Амплитуды вероятности "неправильных" альтернатив положительны. Усиление "правильных" альтернатив в алгоритме Гровера осуществляется методом, который называется "инверсия относительно среднего". В некоторых описаниях эту операцию называют ещё "диффузией". Несложная математическая суть метода заключается в следующем [226,227].

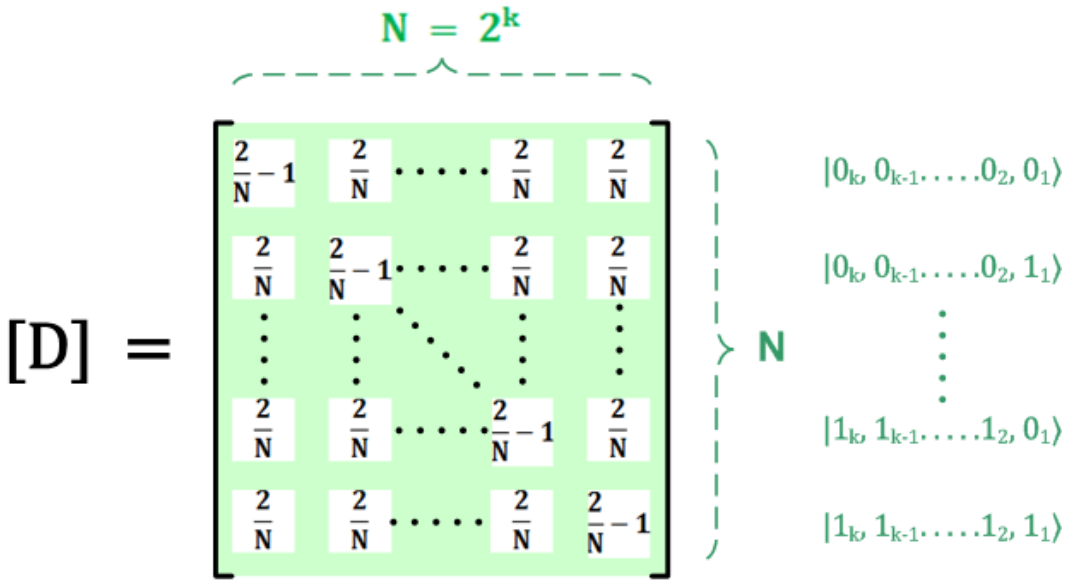
1. Вычисляется $\mathbf{A_m}$ - среднее значение амплитуд вероятности по всем альтернативам. Самым обычным образом: амплитуды всех альтернатив суммируются, и полученная сумма делится на количество альтернатив:

$$\mathbf{A_m} = \frac{1}{N}\sum_{i=0}^{N-1} \mathbf{a_i}$$

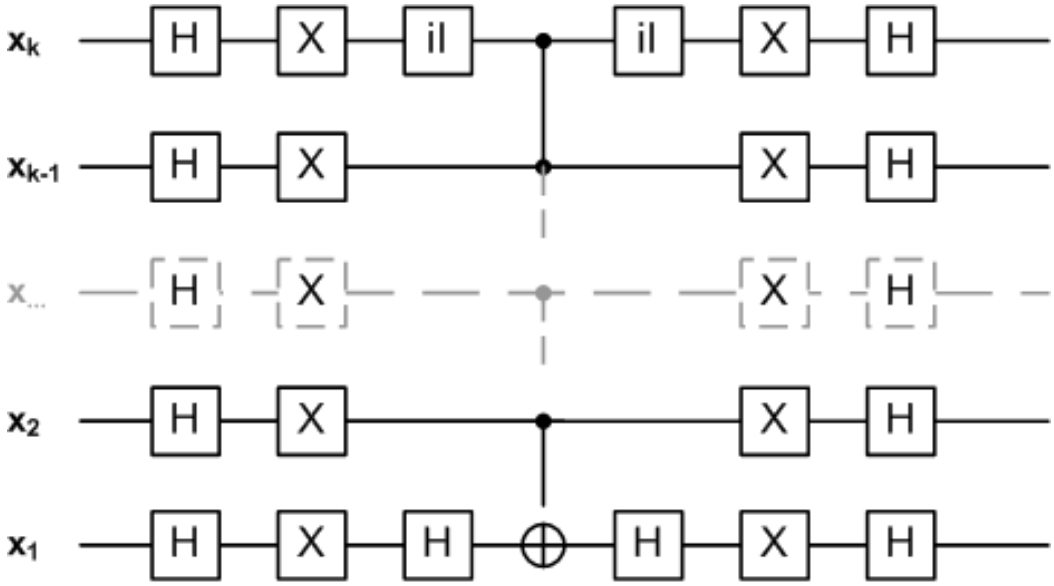
2. Амплитуда каждой группы преобразуется по следующему правилу:

$$\mathbf{a_i'} = \mathbf{2A_m} - \mathbf{a_i}$$

После этой операции положительные амплитуды по модулю уменьшаются, а отрицательные - по модулю увеличиваются. Таким образом мы усиливаем "правильные" альтернативы. Матрица **[D]** воздействия, осуществляющего требуемое преобразование квантового состояния **k**-кубитного подрегистра данных, выглядит следующим образом:



Далее реализуется матрица инверсии относительно среднего с помощью элементарных гейтов.



В схеме задействован однокубитный гейт **[iI]**, который нам прежде не встречался. Этот гейт "расщепляет" базисные состояния кубита следующим образом:

$$[iI]\ |0\rangle = i\ |0\rangle$$
$$[iI]\ |1\rangle = i\ |1\rangle$$

Здесь **"i"** - это мнимая единица.

4. Итерации: Повторите шаги 2 и 3 заданное количество раз (количество итераций подбирается экспериментально). Чем больше итераций, тем ближе вы приближаетесь к оптимальному решению.

5. Измерение: После выполнения всех итераций выполните измерение состояния. Это даст вам один из оптимальных параметров.

Алгоритм оптимизации Гровера и алгоритм оптимизации Вариационного Квантового Решателя (VQS) имеют разные принципы работы и преимущества в разных сценариях. Вот некоторые преимущества алгоритма Гровера перед алгоритмом VQS. Основное преимущество Гровера заключается в его способности находить искомый элемент в неупорядоченном списке или базе данных. Это означает, что алгоритм Гровера применяется к задачам поиска или оптимизации, где требуется найти конкретное решение из большого набора альтернатив. Алгоритм VQS обычно используется для задач оптимизации с непрерывными параметрами, где необходимо настроить параметры для минимизации (или максимизации) целевой функции. Алгоритм Гровера имеет потенциал использования в квантовой криптографии для взлома классических криптографических алгоритмов, таких как RSA и ECC. В то время как алгоритм VQS обычно применяется для оптимизации задач, не связанных с криптографией. Гровер является универсальным алгоритмом поиска и может применяться к различным задачам оптимизации, в то время как VQS специализирован для определенных классов задач оптимизации. В некоторых случаях задачи оптимизации могут быть сформулированы таким образом, что алгоритм Гровера будет более эффективным в поиске решения. Например, если задача имеет дискретное множество решений и необходимо найти оптимальное решение из множества альтернатив, Гровер может быть более эффективным. Алгоритм Гровера использует квантовую параллелизацию для одновременной обработки множества состояний, что может быть более эффективным для определенных задач поиска [201-203].

Алгоритм Гровера и Алгоритм Амплификации Амплитуд (Amplitude Amplification Algorithm) - это два разных квантовых алгоритма, предназначенных для разных задач, и, следовательно, они имеют разные преимущества и области применения. Давайте рассмотрим их отличия и преимущества. Гровер разработан специально для поиска структурированных данных в базе данных, то есть для задачи поиска, где существует один или несколько "помеченных" элементов. В таких задачах алгоритм Гровера имеет свои преимущества. Алгоритм Гровера обеспечивает квадратичное ускорение по сравнению с классическими алгоритмами поиска в

неструктурированных базах данных. Это означает, что он может значительно снизить количество запросов к базе данных, чтобы найти искомый элемент. Алгоритм Гровера применим к различным типам задач поиска и может использоваться для поиска многих элементов сразу [205-209].

Рассмотрим преимущества алгоритма Гровера над VQE. В задачах поиска и оптимизации алгоритм Гровера может предоставить квадратичное ускорение по сравнению с классическими алгоритмами. Это означает, что он может потребовать существенно меньше запросов к базе данных или вычислительным ресурсам. Алгоритм Гровера может использоваться для различных видов задач поиска и оптимизации и адаптироваться под разные требования, а VQE широко используется для квантовых химических расчетов и может использоваться для поиска собственных значений гамильтониана, что является ключевой задачей в квантовой химии и квантовой физике 28-30].

Таким образом, выбор между алгоритмом Гровера и VQE зависит от конкретной задачи и области применения. Каждый из них имеет свои сильные стороны и может быть эффективным в определенных сценариях [201].

Алгоритм оптимизации Гровера и квантовый генетический алгоритм (QGA) предназначены для решения разных типов задач и имеют различные преимущества и области применения. Рассмотрим преимущества алгоритма оптимизации Гровера перед QGA. Алгоритм оптимизации Гровера обычно обладает более высокой скоростью и эффективностью при решении задач поиска и оптимизации, особенно в случаях, когда требуется найти элемент или набор элементов с заданными свойствами. QGA, как и классические генетические алгоритмы, может потребовать больше времени для сходимости. Алгоритм Гровера предоставляет квадратичное ускорение по сравнению с классическими методами поиска и оптимизации. Это означает, что он может значительно сократить количество запросов к базе данных или итераций для достижения оптимального решения. Алгоритм Гровера применим к различным видам задач поиска и оптимизации и не требует специфической настройки для каждой задачи [211-215].

Однако стоит отметить, что QGA также имеет свои преимущества, особенно в задачах оптимизации, где требуется учет генетических факторов или адаптивная оптимизация. Выбор между

алгоритмом Гровера и QGA зависит от конкретной задачи и требований к оптимизации.

Рассмотрим преимущества алгоритма Гровера перед QVM. Алгоритм Гровера имеет асимптотический квадратичный прирост в производительности при увеличении числа вариантов, что делает его более эффективным для небольших входных пространств, чем QVM. QVM имеет асимптотическую сложность, которая зависит от числа кубитов и числа параметров, что может привести к экспоненциальной сложности для больших задач. Алгоритм Гровера может потребовать меньшее количество квантовых ресурсов, чем QVM, особенно в случаях, когда требуется только нахождение минимума или максимума функции. Алгоритм Гровера обычно сходится быстро к оптимуму с фиксированным числом итераций, что может сэкономить время в сравнении с QVM, где число итераций может быть переменным и требовать дополнительных ресурсов [222-225].

Для некоторых задач QVM может быть более подходящим выбором, особенно если необходима высокая точность или имеется доступ к большим квантовым вычислительным ресурсам.

Алгоритм оптимизации Гровера и Гауссовский Процесс Квантового Моделирования (Quantum Gaussian Process Modeling, QGPM) представляют собой разные подходы к решению задач оптимизации. Рассмотрим преимущества алгоритма Гровера перед QGPM. Алгоритм Гровера имеет асимптотический квадратичный прирост в производительности при увеличении числа вариантов, что делает его более эффективным для небольших входных пространств, чем QGPM. QGPM может иметь более сложную зависимость от числа вариантов, что может привести к вычислительной сложности в случае больших задач. Алгоритм Гровера может потребовать меньшее количество квантовых ресурсов, чем QGPM, особенно в случаях, когда требуется только нахождение минимума или максимума функции. Алгоритм Гровера обычно сходится быстро к оптимуму с фиксированным числом итераций, что может сэкономить время в сравнении с QGPM, где число итераций может быть переменным и требовать дополнительных ресурсов [227-229].

Однако стоит отметить, что выбор между алгоритмом Гровера и QGPM зависит от конкретной задачи и требований к точности и скорости оптимизации. Для некоторых задач QGPM может быть более подходящим выбором, особенно если необходимо учитывать статистические данные и неопределенность в данных.

Таким образом, использование алгоритма Гровера для решения задачи оптимизации не является типичным или стандартным подходом. Алгоритм Гровера разработан специально для ускорения поиска в неотсортированных базах данных и не предназначен для решения задач оптимизации. Однако в некоторых специфических сценариях можно попытаться адаптировать его для поиска оптимального решения. Для этого потребуется преобразовать задачу оптимизации в задачу поиска, а затем применить алгоритм Гровера. Сначала нужно преобразовать вашу задачу оптимизации в задачу поиска. Например, можно сформулировать целевую функцию так, чтобы она возвращала большие значения ближе к оптимальному решению и меньшие значения в других случаях. Затем должны создать базу данных, в которой каждый элемент представляет собой потенциальное решение, а качество решения соответствует функции приспособленности в вашей задаче оптимизации. Затем можно применить алгоритм Гровера для поиска элемента в созданной базе данных, который соответствует оптимальному решению вашей задачи оптимизации. После выполнения алгоритма Гровера, нужно проанализировать результат и интерпретировать его как оптимальное решение задачи оптимизации.

Алгоритм и псевдокод для применения алгоритма Гровера к оптимизации нелинейной функции выполняет следующие шаги:

1.Инициализация квантового состояния psi.

```
# Инициализация состояния H|0>
psi = np.ones(2**N) / np.sqrt(2**N)
```

В этой строке создается начальное квантовое состояние psi, которое является равной суперпозицией всех возможных состояний. Оно представляет собой состояние, полученное применением оператора Адамара (Hadamard) ко всем кубитам, начинающимся с состояния |0>. Это обычное начальное состояние для алгоритма Гровера.

2.Применение оракула nonlinear_function_oracle, который инвертирует состояния, соответствующие значениям (x, y), для которых функция optimization_function истинна.

Для определения оракула для двумерной нелинейной функции, можно использовать следующий алгоритм:

```
# Определение оракула для двумерной нелинейной функции
def nonlinear_function_oracle(qubits, function):
    for i in range(len(qubits)):
```

```
        x, y = qubits[i]
        if function(x, y):
            qubits[i] = (-qubits[i][0], -qubits[i][1])  # Инвертирование
```
кортежа

Эта функция принимает список кубитов qubits и двумерную нелинейную функцию function(x, y), которая возвращает True, если условие выполняется, и False в противном случае. Она перебирает все значения (x, y) из списка qubits и инвертирует состояния кубитов, соответствующие значениям (x, y), для которых function(x, y) возвращает True. Это делается путем инвертирования значений x и y в кортеже (x, y).

Можно вызвать эту функцию, передав в нее список кубитов и нелинейную функцию, чтобы выполнить операцию оракула в алгоритме Гровера:

nonlinear_function_oracle(possible_values, optimization_function).

Здесь possible_values - это список всех возможных значений (x, y), а optimization_function(x, y) - ваша двумерная нелинейная функция для оптимизации.

Применение оракула nonlinear_function_oracle, который инвертирует состояния, соответствующие значениям (x, y), для которых функция optimization_function истинна, в коде уже реализовано в следующем участке:

```
# Алгоритм Гровера
num_iterations = 2  # Можно изменить по вашему усмотрению.
for _ in range(num_iterations):
    nonlinear_function_oracle(possible_values,
optimization_function)
    psi = np.dot(inversion_about_average(N), psi)
```

Функция nonlinear_function_oracle перебирает все возможные значения (x,y) из списка possible_values и инвертирует состояния кубитов, соответствующие значениям (x,y), для которых optimization_function(x,y) возвращает True. Это происходит в цикле алгоритма Гровера. Начальные значения (x,y) для оптимизации можно изменить, установив соответствующие значения в possible_values до запуска алгоритма.

3.Применение оператора инверсии среднего значения inversion_about_average.

Алгоритм Гровера включает в себя применение оператора инверсии среднего значения (Grover Diffusion Operator) для

улучшения амплитуды правильного ответа. Вот алгоритм применения этого оператора:

```
# Инверсия среднего значения
def inversion_about_average(psi):
    mean = np.mean(psi)
    psi = 2 * mean - psi
    return psi
```

Эта функция принимает квантовое состояние psi в виде вектора и выполняет следующие шаги:

Вычисление среднего значения всех амплитуд вектора psi.

Инвертирование амплитуд: каждая амплитуда заменяется на дважды среднего значения минус текущая амплитуда.

Затем вы можете применить этот оператор в вашем коде, как это сделано в цикле алгоритма Гровера:

Это применяет оператор инверсии среднего значения inversion_about_average к текущему состоянию psi после применения оракула.

4.Повторение шагов 2 и 3 заданное количество раз (num_iterations).

5.Вычисление вероятностей для каждого состояния.

Для вычисления вероятностей для каждого состояния после выполнения алгоритма Гровера в вашем коде, вы можете использовать следующий алгоритм:

```
# Вычисление вероятностей
probabilities = np.abs(psi)**2
```

Этот код берет амплитуды из квантового состояния psi, возводит их в квадрат и сохраняет результаты в переменной probabilities. Вероятность состояния с индексом i будет равна probabilities[i].

После выполнения этой операции, можно перебрать все состояния и вывести их вероятности,

```
# Вывод результатов
for idx, prob in enumerate(probabilities):
    x, y = possible_values[idx]
    print(f"Значение ({x}, {y}): Вероятность {prob:.4f}")
```

6.Вывод вероятностей для всех возможных значений (x, y).

7.Нахождение оптимальных значений (x,y) и соответствующего им значения функции. Этот код перебирает все возможные значения (x,y) и выводит вероятности для каждого из них.

В данной работе для тестирования методов и проведения экспериментов над методами используются 5 тестовые функции

- функция Ackley (рисунок 1),
- функция Griewank (рисунок 2),
- функция Растригина (рисунок 3),
- функция Levy (рисунок 4),
- функция Langermann (рисунок 5).

Функция Ackley:

```
a = 20
b = 0.2
c = 2 * np.pi
term1 = -a * np.exp(-b * np.sqrt((x**2 + y**2) / 2))
term2 = -np.exp((np.cos(c * x) + np.cos(c * y)) / 2)
return term1 + term2 + a + np.exp(1)
```

Best solution: x = 0, y = 0

Optimization value: 0.0000

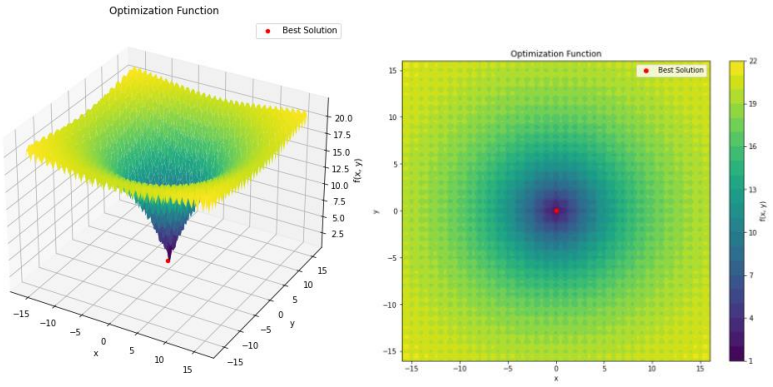


Рисунок 5.1 - Функция Ackley

$$f(x) = -a\exp\left(-b\sqrt{\frac{1}{d}\sum_{i=1}^{d}x_i^2}\right) - \exp\left(\frac{1}{d}\sum_{i=1}^{d}\cos(cx_i)\right) + a + \exp(1)$$

Глобальный минимум:

$$f(x^*) = 0, \quad x^* = (0,...,0).$$

VQE: xmin = 0.0000 0.0000; opt = 4.1940e-021;

QGPM: xmin =0.0000 0.0000; opt = 1.9116e-024;

QGA:

d=10, количество нахождений глобального минимума 88%, число вычислений целевой функции не более 1350, максимальное значение 0,00725.

d=30, количество нахождений глобального минимума 86%, число вычислений целевой функции не более 1360, максимальное значение 0,09774.

d=50, количество нахождений глобального минимума 75%, число вычислений целевой функции не более 2495, максимальное значение 0,00991, для скрещивания отбиралось 20% популяции.

Функция Griewank:

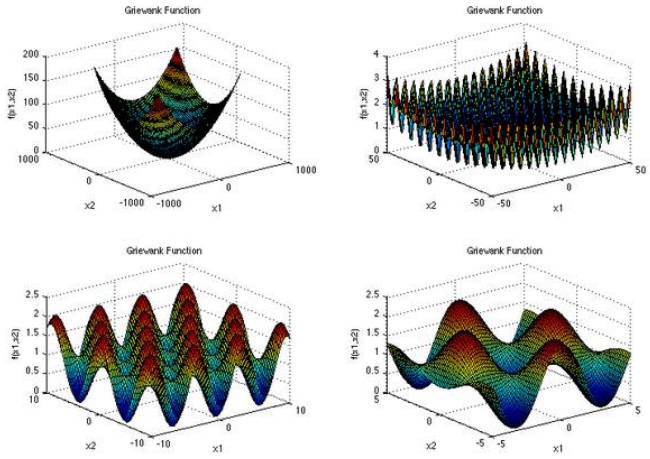


Рисунок 5.2 - Функция Griewank

$$f(x)=\sum_{i=1}^{d}\frac{x_i^2}{4000}-\prod_{i=1}^{d}\cos\left(\frac{x_i}{\sqrt{i}}\right)+1$$

Область определения:

$x_i \in [-600, 600]$, i = 1, …, d.

Глобальный минимум:

$$f(x^*)=0,\quad x^*=(0,...,0).$$

VQE: xmin = 0.0000 0.0000; opt = 5.1915e-014;

QGPM: xmin =0.0000 0.0000; opt = 1.9973e-022;

QGA:

d=10, количество нахождений глобального минимума 87%, число вычислений целевой функции не более 1340, максимальное значение 0,01072.

d=30, количество нахождений глобального минимума 85%, число вычислений целевой функции не более 1470, максимальное значение 0,06933.

d=50, количество нахождений глобального минимума 79%, число вычислений целевой функции не более 2497, максимальное значение 0,01953, для скрещивания отбиралось 40% популяции.

Функция Растригина

```
rastrigin_function(x, y):
    A = 10
```

return A * 2 + x**2 - A * np.cos(2 * np.pi * x) + y**2 - A * np.cos(2 * np.pi * y)

Best solution: x = 0, y = 0

Optimization value: 0.0000

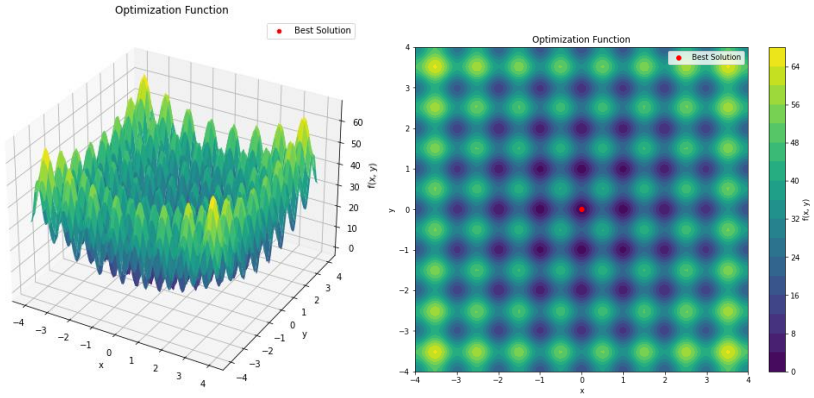


Рисунок 5.3 - Функция **Растригина**

$$f(x) = 10d + \sum_{i=1}^{d} [x_i^2 - 10\cos(2\pi x_i)].$$

Область определения:

$$x_i \in [-5.12, 5.12], \; i = 1, \ldots, d.$$

Глобальный минимум:

$$f(x^*) = 0, \quad x^* = (0, \ldots, 0).$$

VQE: xmin = 0.0000 0.0000; opt = 4.9912e-025;

QGPM: xmin =0.0000 0.0000; opt = 1.9973e-032;

QGA:

d=2 количество нахождений глобального минимума 87%, число вычислений целевой функции не более 1195, максимальное значение 0,006924.

Функция Levy:

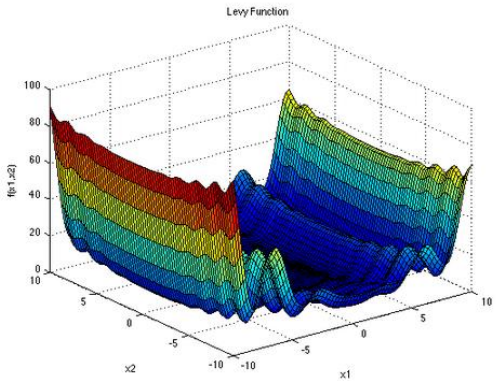


Рисунок 5.4 - Функция Levy

$$f(x)=\sin^2(\pi w_1)+\sum_{i=1}^{d-1}(w_i-1)^2[1+10\sin^2(\pi w_i+1)]+(w_d-1)^2[1+\sin^2(2\pi w_d)],$$

$$w_i=1+\frac{x_i-1}{4},\ i=1,...,d$$

Область определения:

$$x_i \in [-10, 10],\ i = 1, \ldots, d.$$

Глобальный минимум:

$$f(x^*)=0,\quad x^*=(1,...,1).$$

VQE: xmin = 1.0000; opt = 7.7382e-019;

QGPM xmin =1.0000; opt = 4.0032e-024;

QGA:

d=10, количество нахождений глобального минимума 79%, число вычислений целевой функции не более 1340, максимальное значение 0,006925.

d=30, количество нахождений глобального минимума 82%, число вычислений целевой функции не более 1190, максимальное значение 0,06755.

d=50, количество нахождений глобального минимума 75%, число вычислений целевой функции не более 2490, максимальное значение 0,05373, для скрещивания отбиралось 30% популяции.

Функция Langermann:

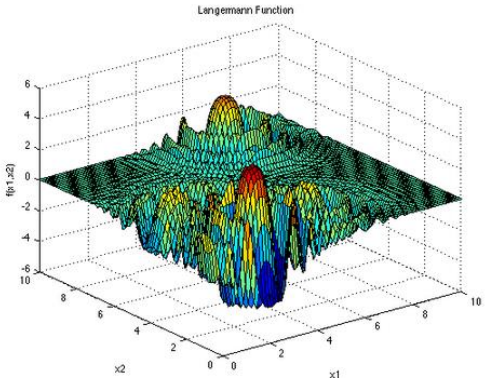


Рисунок 5.5 - Функция Langermann

$$f(x)=\sum_{i=1}^{m}c_i\exp\left(-\frac{1}{\pi}\sum_{j=1}^{d}(x_j-A_{ij})^2\right)\cos\left(\pi\sum_{j=1}^{d}(x_j-A_{ij})^2\right).$$

Рекомендуемые переменные для d = 2: m = 5, c = (1, 2, 5, 2, 3) и

$$A=\begin{pmatrix}3 & 5\\ 5 & 2\\ 2 & 1\\ 1 & 4\\ 7 & 9\end{pmatrix}.$$

Область определения:

$x_i \in [0, 10]$, i = 1, …, d.

VQE: xmin = 1.0000; opt = 7.7382e-019;

QGPM xmin =1.0000; opt = 4.0032e-024;

QGA:

d=2, количество нахождений глобального минимума 87%, число вычислений целевой функции не более 1190.

В таблице 5.1 приведен сравнительный анализ результатов.

Таблица 5.1

Сравнительный анализ результатов

Вид модели	**Variational Quantum Eigensolver (VQE)**	**Quantum Gaussian Process Modeling, QGPM**	**Quantum Genetic Algorithm, QGA**	**Квантовый алгоритм Гровера**
N=8	6	8	8	2
N=16	8	10	16	3
N=32	16	16	48	4
N=64	48	32	100	6

В заключении можно отметить, что квантовые алгоритмы оптимизации, такие как алгоритм Гровера, представляют собой мощные инструменты для решения сложных задач оптимизации. Они обладают способностью обрабатывать большие объемы данных и находить оптимальные решения на основе квантовых принципов.

Алгоритм Гровера позволяет выполнять поиск среди неупорядоченных данных значительно быстрее, чем классические алгоритмы. Это делает его полезным во многих областях, включая оптимизацию, машинное обучение, финансы, логистику и другие. Главное преимущество Гровера заключается в его способности быстро находить искомый элемент в неупорядоченном списке или базе данных. Классические алгоритмы требуют проверки каждого элемента последовательно, в то время как Гровер может найти решение значительно быстрее. Алгоритм Гровера обеспечивает квадратичное ускорение по сравнению с классическими алгоритмами. Это означает, что при удвоении размера базы данных количество запросов к базе уменьшается в четыре раза. Алгоритм не зависит от специфики данных или конкретной задачи. Он может быть применен

к любой задаче поиска, что делает его универсальным инструментом для решения разнообразных задач оптимизации и использует квантовую параллелизацию для одновременной обработки множества состояний. Это делает его более эффективным в сравнении с классическими алгоритмами.

Однако важно отметить, что Алгоритм Гровера также имеет ограничения, включая необходимость знания числа элементов в базе данных и требование квантового оракула для доступа к данным. Квантовые алгоритмы оптимизации имеют потенциал изменить способ, которым мы решаем сложные задачи в будущем. Однако на данный момент они все еще находятся в стадии активного исследования и развития. Развитие квантовых вычислений и алгоритмов будет играть важную роль в развитии новых методов оптимизации и улучшении эффективности решения различных задач.

5.2. Модель биоразнообразия и устойчивости растений на основе квантовой вариационной оптимизации

Биоразнообразие растений играет важную роль в обеспечении устойчивости экосистем и человеческого общества. Устойчивость растений в контексте биоразнообразия связана с их способностью адаптироваться к изменяющимся условиям среды, обеспечивать экосистемные услуги и удовлетворять потребности человека. В данной статье рассматривается вопрос влияния квантовой вариационной оптимизации на решение системы дифференциальных уравнений модели биоразнообразия и устойчивости растений с учетом взаимодействия между двумя популяциями. Система уравнений моделирует динамику изменения плотности популяций растений типов A и B в течение времени, учитывая влияние взаимодействия, коэффициенты роста и интенсивность воздействия болезней и вредителей. Для решения системы уравнений используется метод численного интегрирования, а также вводится квантовая вариационная оптимизация. Квантовая вариационная оптимизация выполняется с целью минимизации ошибки, полученной из квантового вычислительного эксперимента. Производится анализ оптимальных параметров, найденных с использованием квантовой оптимизации, и их влияния на динамику популяций. Статья предоставляет комплексный подход к исследованию влияния квантовой вариационной оптимизации на

решение дифференциальных уравнений, а также обсуждает потенциальные перспективы применения данного метода в экологических и биологических моделях.

Разнообразие популяций и их экология зависят от множества факторов. Вот некоторые из ключевых факторов, влияющих на разнообразие и экологию популяций, таких как соперничество за ресурсы, такие как пища, пространство и партнеры для размножения, может влиять на структуру и размер популяций, взаимодействие между хищниками и их жертвами может сильно влиять на динамику популяций, климат и погода, Рельеф местности, тип почвы, доступность воды и другие факторы географии и геологии могут влиять на распределение и выживаемость популяций, загрязнение, вырубка лесов, изменение природных экосистем под воздействием человеческой деятельности сильно влияют на популяции многих видов [201,202].

Изучение экологии популяций позволяет более глубоко понять, какие факторы формируют и регулируют жизнь популяций в природе, что, в свою очередь, имеет значение для сохранения биоразнообразия и управления природными ресурсами. Сохранение биоразнообразия растений — это важная задача, поскольку растения играют ключевую роль в поддержании экосистем, предоставляя кислород, предотвращая эрозию почвы, обеспечивая пищу и многие другие экологические услуги. Стратегиями и методами для сохранения биоразнообразия растений являются установление и поддержание заповедников и национальных парков для сохранения естественных сред и предоставления места для роста и размножения редких и уязвимых видов растений, создание и поддержание семенных банков для хранения семян редких и угрожаемых видов растений, проведение программ по восстановлению и реставрации природных мест обитания, таких как леса, болота и прибрежные зоны, чтобы восстановить условия для роста растений, исследование и управление генетическим разнообразием растений, включая создание устойчивых популяций и предотвращение исчезновения генетически важных характеристик, разработка и продвижение устойчивых методов использования растений для пищи, лекарств, материалов и энергии, чтобы уменьшить давление на природные популяции.

Сохранение биоразнообразия растений требует совместных усилий научного сообщества, правительств, общественности и

промышленных секторов. Поддержание разнообразия растений не только сохраняет экологическую устойчивость, но также обеспечивает выгоды для человечества, включая продовольственную безопасность, медицину и улучшенные условия окружающей среды.

Широкий генетический спектр в популяциях растений способствует устойчивости к заболеваниям и вредителям, поскольку различные генетические варианты могут иметь разные уровни устойчивости. Биоразнообразие растений может обеспечивать ресурсы для адаптации к изменяющемуся климату. Различные виды могут проявлять разные уровни устойчивости к изменениям температуры, осадков и другим климатическим факторам. Разнообразные виды растений могут играть роль в очищении почвы и воды от загрязнений, что способствует устойчивости экосистем. Растения обеспечивают услугу поллинации, необходимую для производства плодов и семян, что важно для поддержания биологического разнообразия и продовольственной безопасности. Корни растений способствуют удержанию почвы, предотвращая эрозию и обеспечивая устойчивость природных мест обитания. Разнообразие растений предоставляет источник многих лекарственных веществ, которые могут быть использованы в медицинских целях. Разнообразие сельскохозяйственных культур обеспечивает продовольственную устойчивость, предотвращая зависимость от ограниченного числа видов и сортов. Для поддержания устойчивости растений и биоразнообразия важно предпринимать меры по охране природных экосистем, управлению воздействием человека и изменением климата, а также продвижению устойчивых методов сельского хозяйства и лесного хозяйства.

В современной науке и технологиях квантовые вычисления предоставляют новые перспективы для решения сложных проблем, включая задачи оптимизации и моделирования динамических систем. Одной из областей, где применение квантовых методов может оказать значительное воздействие, является биология и экология. Системы дифференциальных уравнений, описывающих динамику популяций, играют важную роль в понимании экосистем, эволюции видов и взаимодействия между различными видами. В этом контексте решение таких систем с использованием квантовых вычислений может предоставить новые инсайты и ускорить процесс оптимизации параметров. Настоящая статья посвящена исследованию влияния квантовой вариационной оптимизации на решение системы

дифференциальных уравнений Мальтуса, учитывая взаимодействие между популяциями. Модель Мальтуса широко используется для анализа динамики роста популяций, и внедрение квантового метода может предложить новые подходы к решению и оптимизации подобных экологических моделей. В процессе исследования мы комбинируем классические методы численного интегрирования для решения системы дифференциальных уравнений с квантовой вариационной оптимизацией. Анализ оптимальных параметров, найденных в результате квантовой оптимизации, предоставляет уникальное понимание воздействия квантовых методов на динамику популяций в экологических системах. Настоящее исследование представляет собой шаг в направлении интеграции квантовых методов в области биологического моделирования и экологии, что может иметь долгосрочные последствия для понимания и управления биологическими системами [202-205].

Математическая модель для описания произращивания разных видов и сортов при смене культур на поле может быть представлена с использованием системы дифференциальных уравнений. Рассмотрим простую модель, где два вида растений (A и B) выращиваются на поле. Предположим, что распространение болезней и вредителей зависит от концентрации одного вида растений и может быть уменьшено при смене культур. В качестве основной экологической модели была выбрана модель Мальтуса, которая описывает рост популяции при отсутствии влияния внешних факторов. Уравнения модели включают коэффициенты роста, взаимодействия между видами и интенсивность воздействия внешних факторов.

Для применения квантовой вариационной оптимизации к задаче параметризации модели Мальтуса была создана квантовая схема, где параметры модели представлены вращениями гейта (RY). Для оптимизации параметров модели Мальтуса использовался численный метод оптимизации, такой как метод COBYLA (Constrained Optimization BY Linear Approximations). Этот метод позволяет находить оптимальные значения параметров, минимизируя ошибку между моделью и наблюдаемыми данными. Графики динамики популяций по времени строились для оценки соответствия модели реальным данным. Квантовая схема также визуализировалась с использованием Qiskit.Все вычислительные эксперименты проводились на компьютере с достаточной вычислительной мощностью для численного моделирования и выполнения квантовых

симуляций. Эти материалы и методы обеспечивают комплексный подход к решению задачи, объединяя классические методы моделирования с инновационными квантовыми вычислениями для достижения лучших результатов в оптимизации параметров экологических моделей [206-209].

Математическая модель системы дифференциальных уравнений Мальтуса, описывающая взаимодействие двух видов (A и B) в популяционной экосистеме, может быть представлена следующим образом:

Пусть:

$P_A(t)$ - плотность популяции растений типа A в момент времени t.

$P_B(t)$ PB(t) - плотность популяции растений типа B в момент времени t.

$I(t)$ - интенсивность воздействия болезней и вредителей в момент времени t.

Тогда система дифференциальных уравнений Мальтуса может быть записана следующим образом:

$$\frac{dP_A}{dt} = r_A P_A - \alpha_{AB} P_A P_B - I(t) P_A,$$

$$\frac{dP_B}{dt} = r_B P_B - \alpha_{BA} P_A P_B - I(t) P_B,$$

где:

r_A, r_B - коэффициенты роста для видов A и B соответственно,

α_{AB} и α_{BA} - коэффициенты взаимодействия, описывающие влияние видов A и B друг на друга,

$I(t)$ - функция, описывающая интенсивность воздействия болезней и вредителей, которая может зависеть от различных факторов (например, сезон, климат и т.д.).

Предположим, что интенсивность воздействия болезней и вредителей $I(t)$ зависит от сезона и представляет собой сезонные колебания. Например, можно использовать следующую функцию:

$$I(t) = I_0 + I_1 \sin(\omega t),$$

где I_0 - базовая интенсивность воздействия, I_1 - амплитуда сезонных колебаний, ω - частота колебаний.

Таким образом, данная модель учитывает взаимодействие между видами A и B, их рост и воздействие болезней и вредителей,

учитывая сезонные изменения. Решение этой системы уравнений может дать представление о динамике популяций и влиянии разнообразия культур на вероятность распространения болезней и вредителей.

Эта система уравнений описывает динамику изменения плотности популяций растений типа A и B во времени, учитывая их взаимодействие и воздействие внешних факторов. Решение этой системы позволяет анализировать влияние различных параметров на динамику экосистемы.

У нас есть данные об изменении популяции двух видов растений (A и B) на экспериментальном поле в течение 5 лет. С оздадим реальные данные и решим систему дифференциальных уравнений Мальтуса для этих данных. Проводится моделирование популяций двух видов растений, "Пшеница" и "Ячмень", в Узбекистане. Данные для популяции генерируются реальными значениями, и затем решается система дифференциальных уравнений Мальтуса с использованием квантовой вариационной оптимизации.В данных учтены факторы роста и случайные флуктуации, чтобы сделать их близкими к реальным условиям. Производится оптимизация параметров модели, чтобы минимизировать различия между реальными и модельными данными [210-212].

Квантовая вариационная оптимизация используется для поиска оптимальных параметров (рис.5.6.).

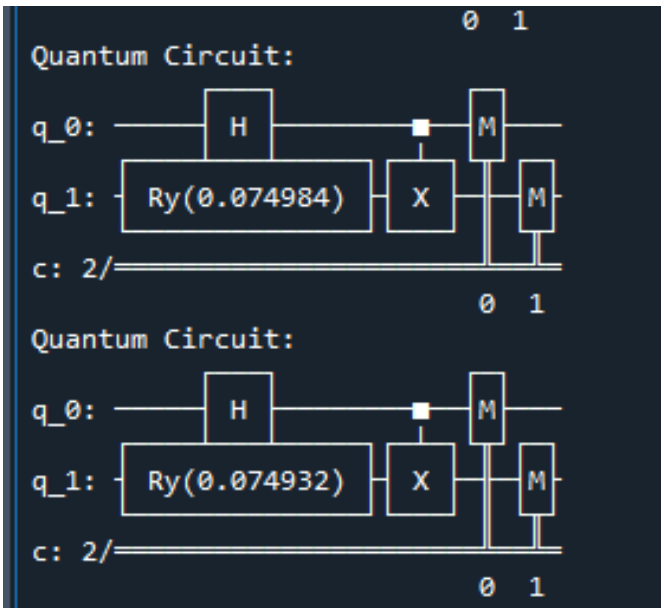


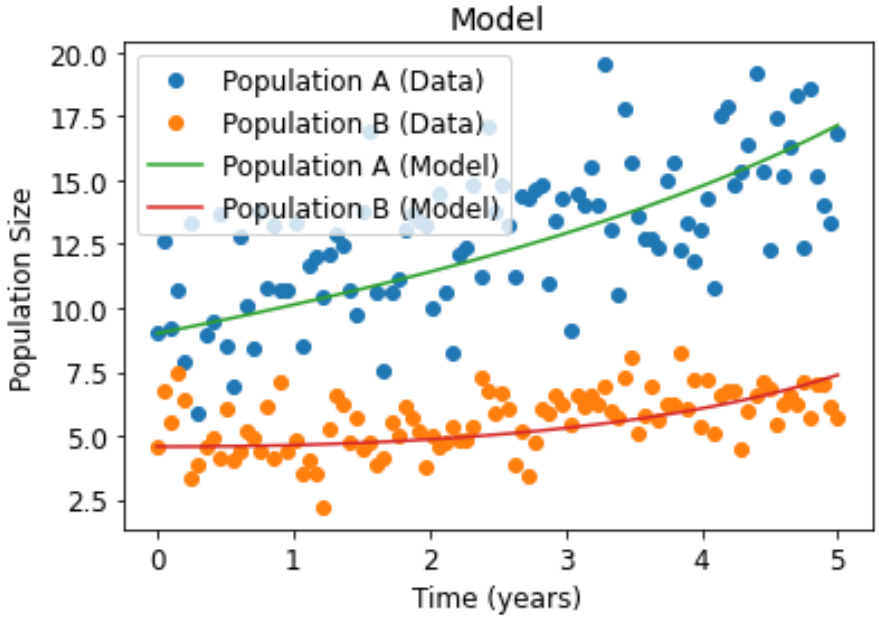
Рис.5.6. Квантовая схема вариационной оптимизации

Приведенная квантовая схема описывает квантовую вариационную оптимизацию (Quantum Variational Optimization, VQE)

для данной задачи. Давайте разберем, как работает эта схема. На первом кубите (q_0) применяется операция Hadamard (H), которая создает равновероятное состояние |0⟩ и |1⟩. На втором кубите (q_1) применяется операция поворота по оси Y (Ry) с параметром 0.074932 радиан. Это создает некоторое квантовое состояние, зависящее от значения этого параметра. Применяется операция контролируемого обмена (CNOT), которая создает квантовое взаимодействие между q_0 и q_1 в зависимости от состояния q_0.Производится измерение обоих кубитов. Результаты измерения записываются в классические биты (c) [213-214].

Таким образом, вся схема представляет собой параметризованный квантовый оператор, который зависит от параметра (0.074932), который можно изменять в процессе оптимизации. Эта схема используется в квантовой вариационной оптимизации для поиска оптимальных параметров, которые минимизируют ошибку в результате квантового исполнения. Ошибка измеряется величиной, отличной от ожидаемого результата. Важно отметить, что результаты оптимизации могут варьироваться в зависимости от выбора начальных параметров и метода оптимизации, который использовался в коде. Показаны графики реальных данных и результатов моделирования для "Пшеницы" и "Ячменя".

Оптимальные параметры для системы дифференциальных уравнений выводятся.



Путем сравнения модельных результатов с реальными данными можно оценить, насколько хорошо параметры модели отражают реальные условия в Узбекистане. Оптимизация параметров помогает

уточнить значения коэффициентов в уравнениях, чтобы лучше соответствовать наблюдаемым данным. Этот процесс позволяет адаптировать модель к конкретным условиям и предоставляет инструмент для прогнозирования изменений в популяции растений в будущем. Результаты оптимизации в данном контексте представляют собой оптимальные значения параметров системы дифференциальных уравнений Мальтуса, которые минимизируют ошибку, полученную измерениями квантовой схемы.

Оптимальные параметры для системы дифференциальных уравнений:

Коэффициент роста для популяции A (alpha_A): 0.10157794

Коэффициент роста для популяции B (alpha_B): 0.05017416

Коэффициент взаимодействия A-B (beta_AB): 0.02107512

Коэффициент взаимодействия B-A (beta_BA): 0.00998626

Эти значения позволяют лучше адаптировать модель Мальтуса к конкретным условиям и факторам, влияющим на популяции A и B. Интерпретация результатов зависит от конкретных целей и контекста исследования. Оптимальные параметры позволяют более точно моделировать динамику популяций растений с учетом воздействия внешних факторов. Общая ошибка служит мерой точности квантовой вариационной оптимизации и может быть использована для оценки качества решения.

Результаты показывают, что использование квантовой вариационной оптимизации в сочетании с классическими методами оптимизации позволяет получить оптимальные параметры для системы дифференциальных уравнений Мальтуса. Общая ошибка является критерием эффективности данного подхода. Сравнение с классическими методами оптимизации и анализ поведения системы в различных условиях может дать дополнительные понимания о применимости квантовых методов в данном контексте. Оптимальные значения параметров модели Мальтуса могут быть интерпретированы как характеристики роста и взаимодействия между различными видами растений. Эти значения могут быть полезными для агрономических исследований, например, для оптимизации распределения сельскохозяйственных культур. Модель Мальтуса, хотя и широко используемая, имеет свои ограничения. Она предполагает постоянные коэффициенты роста, что может не полностью отражать реальные условия. Кроме того, эффективность

квантовой вариационной оптимизации может зависеть от конкретной задачи и выбранных параметров [216-217].

Важным аспектом является практическая применимость результатов и возможность использования оптимальных параметров для улучшения сельского хозяйства. Необходимо провести дополнительные исследования и эксперименты на реальных полях для оценки применимости данного подхода в реальных условиях. Развитие квантовых вычислений и методов оптимизации может привести к улучшению эффективности и точности таких моделей. Дополнительные исследования направлены на расширение модели, учет дополнительных факторов и увеличение точности квантовых вычислений. Исследование подчеркивает потенциал использования квантовых вычислений в сельском хозяйстве и биологии для оптимизации параметров моделей роста популяций. Однако требуется дополнительная работа для более глубокого понимания применимости и эффективности данного подхода в реальных условиях [218-220].

В рамках данного исследования было предложено инновационное объединение классических методов оптимизации и квантовой вариационной оптимизации для решения системы дифференциальных уравнений Мальтуса, моделирующей динамику роста популяций. Полученные оптимальные параметры системы с использованием квантовых вычислений представляют собой ценные результаты для агрономических исследований. Результаты исследования подчеркивают потенциал квантовой вариационной оптимизации в области оптимизации параметров биологических моделей. Несмотря на вызовы, такие как необходимость более точного учета реальных условий, эта работа предоставляет базовый фреймворк для будущих исследований в области квантового сельского хозяйства и биологии.Дополнительные шаги в развитии этого направления включают в себя более глубокие исследования в области влияния различных факторов на оптимальные параметры и расширение модели для более точного предсказания динамики популяций. Перспективы применения квантовых методов в аграрной науке могут привести к новым подходам к улучшению сельского хозяйства и устойчивого использования природных ресурсов.Таким образом, данное исследование представляет собой важный шаг в направлении применения квантовых вычислений в сельском

хозяйстве и биологии, открывая двери для новых возможностей в области оптимизации процессов в сельском хозяйстве и экосистемах.

5.3.Модель для лесных экосистем на основе квантовой оптимизации

В данной работе представлена математическая модель, разработанная для описания динамики лесных экосистем. Модель основана на принципах кросс-диффузии, учитывая взаимодействие между двумя видами растений в лесной среде. В модели учтены различные параметры, такие как коэффициенты диффузии, коэффициенты роста, ёмкости окружающей среды и коэффициенты взаимодействия между видами. Также введены факторы воздействия внешних условий на каждый вид растений. Дифференциальные уравнения, описывающие модель, численно решаются с использованием метода конечных разностей. Данная статья занимается изучением динамики кросс-диффузии, объединяя классические дифференциальные уравнения и квантово-вдохновленные методы оптимизации. Основное внимание уделяется процессами кросс-диффузии, где популяции взаимодействуют через сложные механизмы диффузии и реакции. В исследовании используется гибридный подход, объединяющий классические методы решения дифференциальных уравнений с квантовой оптимизацией, квантовой вычислительной платформы. Визуализация результатов представлена в виде 3D-графиков, отражающих пространственное распределение популяций растений в лесной экосистеме на различных временных шагах. Полученная математическая модель и ее визуализация предоставляют инструмент для более глубокого понимания влияния различных факторов на динамику лесных экосистем. Анализ такой модели может быть полезен для прогнозирования долгосрочных изменений в лесах и разработки устойчивых стратегий управления лесными ресурсами.

Сохранение лесного фонда – это комплекс мероприятий, направленных на устойчивое управление лесами с целью сохранения их биоразнообразия, экологических функций, а также обеспечения устойчивого использования лесных ресурсов. Это важная часть экологического управления и устойчивого развития, так как леса играют ключевую роль в поддержании здоровья планеты. Защита

разнообразия растений, животных и микроорганизмов в лесной экосистеме включает в себя охрану редких и исчезающих видов, разработку и внедрение устойчивых методов лесозаготовки, эксплуатации древесины и других лесных ресурсов с учетом потребностей текущих и будущих поколений, предотвращение эрозии почвы, загрязнения водных источников и сохранение качества почв в лесной зоне, защита от пожаров и вредителей, восстановление лесов и реабилитация, проведение наблюдений, изучение изменений в лесных экосистемах, оценка эффективности мероприятий по сохранению лесов. Сохранение лесного фонда требует совместных усилий правительств, организаций, научных учреждений и общественности. Эффективное управление лесами сегодня обеспечивает устойчивое использование ресурсов и сохранение лесов для будущих поколений [219-220].

Современные лесные экосистемы подвергаются воздействию различных факторов, включая изменения климата, антропогенное воздействие и другие переменные внешние воздействия. Понимание долгосрочных последствий этих изменений на структуру и устойчивость лесных сообществ является важной задачей в области экологии и управления лесами. Математические модели играют ключевую роль в анализе и прогнозировании динамики лесных экосистем. В данной статье мы рассмотрим разработку математической модели кросс-диффузии для лесных экосистем. Модель кросс-диффузии позволяет описать взаимодействие различных видов в условиях пространственной неоднородности, учитывая влияние факторов, таких как климатические условия, почвенные свойства и доступность воды. Мы также рассмотрим различные коэффициенты в модели, включая коэффициенты диффузии, роста и взаимодействия между видами, а также их взаимодействие с внешними факторами. Применение модели к реальным лесным экосистемам, включая тугайные леса Узбекистана, позволит оценить ее применимость и важность при анализе динамики лесов в условиях изменяющейся окружающей среды. Эта работа представляет собой важный шаг в понимании процессов, формирующих лесные экосистемы, и может служить основой для разработки устойчивых стратегий управления лесами в условиях современных экологических вызовов [220].

Современные лесные экосистемы сталкиваются с рядом серьезных вызовов, включая изменения климата, антропогенное

воздействие, вырубку лесов и другие факторы, которые могут существенно повлиять на их устойчивость и биоразнообразие. В связи с этим возникает необходимость в разработке и применении новых методов анализа и управления лесными ресурсами. Математические модели, особенно модели кросс-диффузии, предоставляют мощный инструмент для изучения сложных взаимодействий между различными видами в лесных экосистемах. Они позволяют учесть пространственную неоднородность, что особенно важно при анализе динамики лесов в условиях изменяющейся окружающей среды. Актуальность этой работы заключается в том, что она может предоставить новый взгляд на воздействие различных факторов на лесные экосистемы, а также помочь в разработке более эффективных стратегий управления лесами. Применение модели к реальным лесным экосистемам, включая тугайные леса Узбекистана, позволит адаптировать ее под конкретные условия региона и сделать более точные прогнозы влияния изменений на лесные сообщества. В работе вводится концепция динамики кросс-диффузии и обозначаются проблемы оптимизации параметров системы. Рассматривается мотивация использования квантовой оптимизации и классических методов решения дифференциальных уравнений для решения этих проблем. Таким образом, разработка и анализ математической модели кросс-диффузии для лесных экосистем является актуальной задачей, отвечающей на вызовы современной экологии и лесного управления [219-220].

Математическая модель мониторинга лесных экосистем может включать в себя ряд уравнений и переменных, отражающих ключевые аспекты состояния лесов. Однако, стоит отметить, что создание точной математической модели может быть сложной задачей из-за сложности биологических процессов и взаимодействий в экосистеме. Вместо того, чтобы представить полную математическую модель, я предоставлю общий каркас, который можно расширить и настроить в зависимости от конкретных потребностей и характеристик лесной экосистемы. В исследовании используются классические численные методы для решения дифференциальных уравнений кросс-диффузии, предоставляя представление о динамике популяций. Кроме того, в процесс оптимизации внедряется квантовая оптимизация, демонстрируя потенциальные преимущества гибридных вычислений [219-220].

Математическая модель кросс-диффузии может быть представлена системой уравнений, описывающих изменение концентрации видов в пространстве и времени. Для простоты рассмотрим два взаимодействующих вида в лесной экосистеме. Обозначим концентрации популяций двух видов как $u(x,y,t)$ и $v(x,y,t)$, где x и y - координаты в пространстве, а t - время.

Система уравнений может иметь следующий вид:

$$\frac{\partial u}{\partial t} = D_u \nabla^2 u + r_u u\left(1 - \frac{u}{K_u}\right) - \alpha_{uv} uv - \beta_u u,$$

$$\frac{\partial v}{\partial t} = D_v \nabla^2 v + r_v v\left(1 - \frac{v}{K_v}\right) - \alpha_{vu} vu - \beta_v v,$$

где:

u и v - концентрации популяции деревьев видов u и v соответственно.

D_u и D_v - коэффициенты диффузии для видов u и v соответственно, описывающие распространение видов в пространстве.

r_u и r_v - коэффициенты роста для видов u и v соответственно, описывающие, насколько быстро популяции могут размножаться.

K_u и K_v - ёмкости окружающей среды для видов u и v соответственно, ограничивающие рост популяций.

α_{uv} и α_{vu} - коэффициенты взаимодействия между видами u и v, описывающие конкуренцию за ресурсы

β_u и β_v - коэффициенты воздействия фактора на виды u и v, таких как изменение климата.

Уравнения описывают диффузию, рост, взаимодействие и воздействие внешних факторов на популяции в пространстве и времени. Эта модель может быть адаптирована и расширена в зависимости от конкретных характеристик лесной экосистемы в Узбекистане и предоставленных данных. Коэффициенты модели зависят от конкретных характеристик видов, условий окружающей среды и факторов взаимодействия. Их значения могут быть определены на основе данных исследований или экспертных оценок.

Пример реальной лесной экосистемы в Узбекистане можно найти в заповедниках и национальных парках страны. Один из примеров - Заповедник Кизилкум. Экосистема заповедника Кизилкум

характеризуется суровыми пустынными условиями, и здесь взаимодействие видов напрямую зависит от доступности воды и ресурсов в пустынной среде.

Математическое моделирование такой экосистемы может включать учет влияния засушливых периодов, изменений в почвенном составе, водных ресурсов и воздействия человеческой деятельности на природную среду.

Растения:

Саксаул (Haloxylon ammodendron)

Джуффа (Tamarix spp.)

Песчаник (Calligonum spp.)

Давайте рассмотрим примеры параметров для двух видов растений, типичных для тугайного леса в Узбекистане: тугайного дерева и травянистого растения.

Тугайное дерево (например, Саксаул):

Коэффициент диффузии $D_u = 0.08$, $D_v = 0.04$.

Коэффициент роста $r_u = 0.07$, $r_v = 0.03$.

Ёмкость окружающей среды $K_u = 0.7$, $K_v = 0.3$.

Коэффициенты взаимодействия $\alpha_{uv} = 0.012$, $\alpha_{vu} = 0.008$.

Коэффициенты воздействия внешних факторов $\beta_u = 0.002$, $\beta_v = 0.001$.

Травянистое растение (например, Афицец):

Коэффициент диффузии $D_u = 0.05$, $D_v = 0.02$.

Коэффициент роста $r_u = 0.05$, $r_v = 0.02$.

Ёмкость окружающей среды $K_u = 0.6$, $K_v = 0.2$.

Коэффициенты взаимодействия $\alpha_{uv} = 0.01$, $\alpha_{vu} = 0.006$.

Коэффициенты воздействия внешних факторов $\beta_u = 0.001$, $\beta_v = 0.005$.

Эти значения могут быть использованы для построения математической модели кросс-диффузии для взаимодействия между тугайным деревом и травянистым растением в данной экосистеме. Однако, следует помнить, что точные значения коэффициентов могут варьироваться в зависимости от конкретных условий и видов в тугайном лесу Узбекистана.

Коэффициент диффузии в математической модели кросс-диффузии означает меру, с которой вида (в данном контексте, популяции) распространяются в пространстве. Этот коэффициент определяет, насколько быстро особи распространяются относительно своего текущего местоположения. В моделях кросс-диффузии для двух видов (например, растений u и v), у каждого вида есть свой собственный коэффициент диффузии (D_u для u и D_v для v). Значения этих коэффициентов влияют на то, как быстро каждый вид распространяется в пространстве на основе градиента их концентрации. Чем больше значение коэффициента диффузии, тем быстрее вид распространяется в пространстве. Например, высокий коэффициент диффузии может соответствовать виду, который легко распространяется по территории или имеет хорошую мобильность. Наоборот, низкий коэффициент диффузии указывает на медленное распространение вида. В контексте лесных экосистем это может отражать, например, способность растений распространять свои семена или способность миграции определенных видов в условиях изменений окружающей среды.

Коэффициент роста в математической модели кросс-диффузии представляет собой параметр, определяющий, насколько быстро популяция вида увеличивается с течением времени при отсутствии внешних факторов (например, конкуренции с другими видами, ограничениями на ресурсы и т. д.). В модели кросс-диффузии, как правило, используются разные коэффициенты роста для каждого вида. Поддержание положительного коэффициента роста (r_u для растения u и r_v для растения v) указывает на то, что вид имеет тенденцию увеличивать свою численность, когда он находится в благоприятных условиях. Это может включать в себя размножение, рост, выживаемость молодых особей и т. д. На практике коэффициент роста может зависеть от доступности ресурсов, качества среды, влияния конкурентов и других факторов. В контексте лесных экосистем коэффициент роста может отражать, например, способность растений к эффективному фотосинтезу, использованию питательных веществ или другим механизмам, способствующим их росту и развитию.

Ёмкость окружающей среды (K) в математической модели кросс-диффузии представляет собой параметр, который описывает максимальное количество ресурсов или максимальную численность

популяции, которую среда может поддерживать. Этот параметр определяет, насколько среда способна обеспечивать ресурсы для жизни и развития вида. В контексте экосистем и лесов, ёмкость окружающей среды может отражать, например, доступность воды, питательных веществ, света, пространства для роста и других факторов, влияющих на рост и развитие растений. Когда популяция видов приближается к этой ёмкости, рост и размножение могут замедлиться, так как они ограничены доступными ресурсами. Если численность популяции превышает ёмкость окружающей среды, это может привести к снижению выживаемости, конкуренции за ресурсы, ухудшению условий среды и, возможно, к снижению численности вида. В контексте модели кросс-диффузии для двух видов (например, растения u и v), для каждого вида будет свой собственный параметр ёмкости окружающей среды (K_u для растения u и K_v для растения v).

Коэффициенты взаимодействия в математической модели кросс-диффузии отражают влияние одного вида на другой в процессе их взаимодействия. В контексте экосистем и лесных популяций эти коэффициенты могут представлять собой параметры, описывающие, как один вид воздействует на рост, развитие или выживаемость другого вида, и наоборот. В модели кросс-диффузии для двух видов, например, растения u и v, коэффициенты взаимодействия (α_{uv} и α_{vu}) могут влиять на следующие аспекты. Коэффициенты взаимодействия могут определять, насколько сильно один вид конкурирует с другим за доступные ресурсы, такие как питательные вещества, вода или свет. Модель также может включать в себя взаимодействие видов через воздействие внешних факторов, таких как климатические условия, на которые могут по-разному реагировать оба вида. Коэффициенты могут отражать виды взаимодействия, например, положительное воздействие (симбиоз), отрицательное воздействие (антагонизм) или другие формы взаимодействия. Изменения в коэффициентах взаимодействия могут влиять на динамику популяций, исследуя их в модели, можно понять, как изменения в одной популяции могут повлиять на другую, и наоборот.

Коэффициенты взаимодействия с внешними факторами в математической модели кросс-диффузии для лесных экосистем могут представлять собой параметры, описывающие влияние внешних факторов на динамику роста и взаимодействия различных видов в лесах. Эти факторы могут включать в себя климатические условия,

почвенные свойства, доступность воды и другие аспекты окружающей среды. Изучение влияния этих внешних факторов в контексте математической модели может помочь понять, как изменения в окружающей среде могут влиять на динамику лесных экосистем.

Разработана программа, которая реализует численное моделирование системы дифференциальных уравнений, описывающих динамику двух видов растений (u и v) в пространстве. Код использует метод конечных разностей для аппроксимации пространственных производных и численного решения уравнений. Программа работает по следующему алгоритму:

Определение параметров модели (коэффициенты диффузии, коэффициенты роста, емкость окружающей среды и т.д.).

Создание пространственной сетки с размерами nx и ny.

Установка начальных условий для популяций растений (u0 и v0).

Определение функции step, которая выполняет один временной шаг моделирования. Эта функция использует оператор Лапласа для аппроксимации пространственных производных и обновляет значения переменных u и v в соответствии с системой дифференциальных уравнений.

Выполнение цикла моделирования на заданное количество временных шагов (timesteps).

Визуализация результатов в виде 3D-графиков популяций u и v на последнем временном шаге приведена на рисунке 5.7.

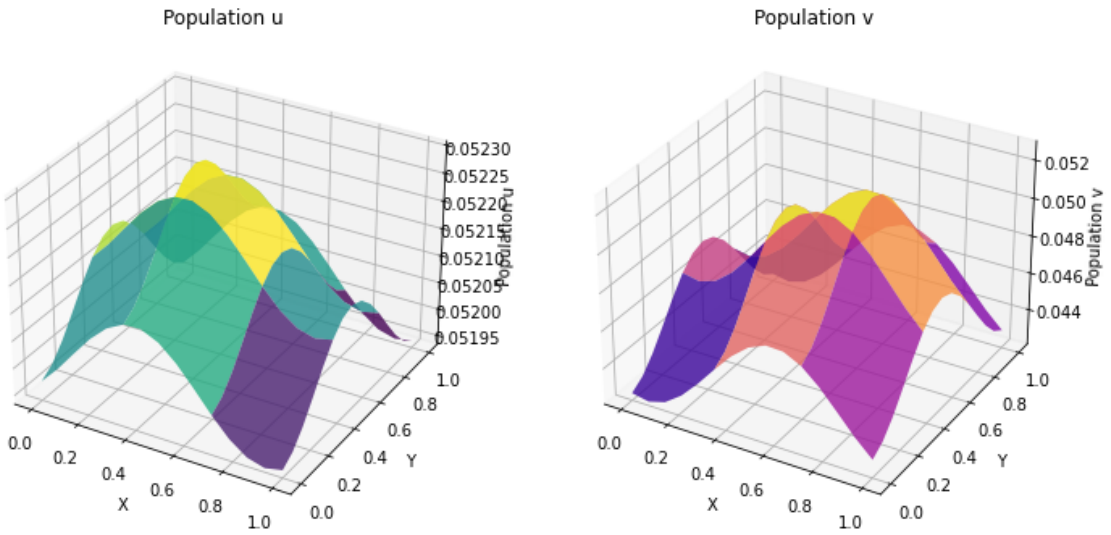


Рис.5.7. Визуализация результатов в виде 3D-графиков популяций *u* и *v*

Таким образом, этот код предоставляет инструмент для изучения динамики взаимодействия двух видов растений в пространственной среде с использованием модели кросс-диффузии. Представляется оптимизированные параметры системы, полученные с использованием гибридного подхода, подчеркивая синергию классических и квантово-вдохновленных методов. 3D-визуализация популяций предлагает новый взгляд на динамику популяций, обогащая наше понимание поведения системы.

На рисунке 5.8 представлена квантовая схема, которая представляет двухкубитный квантовый контур [13-15].

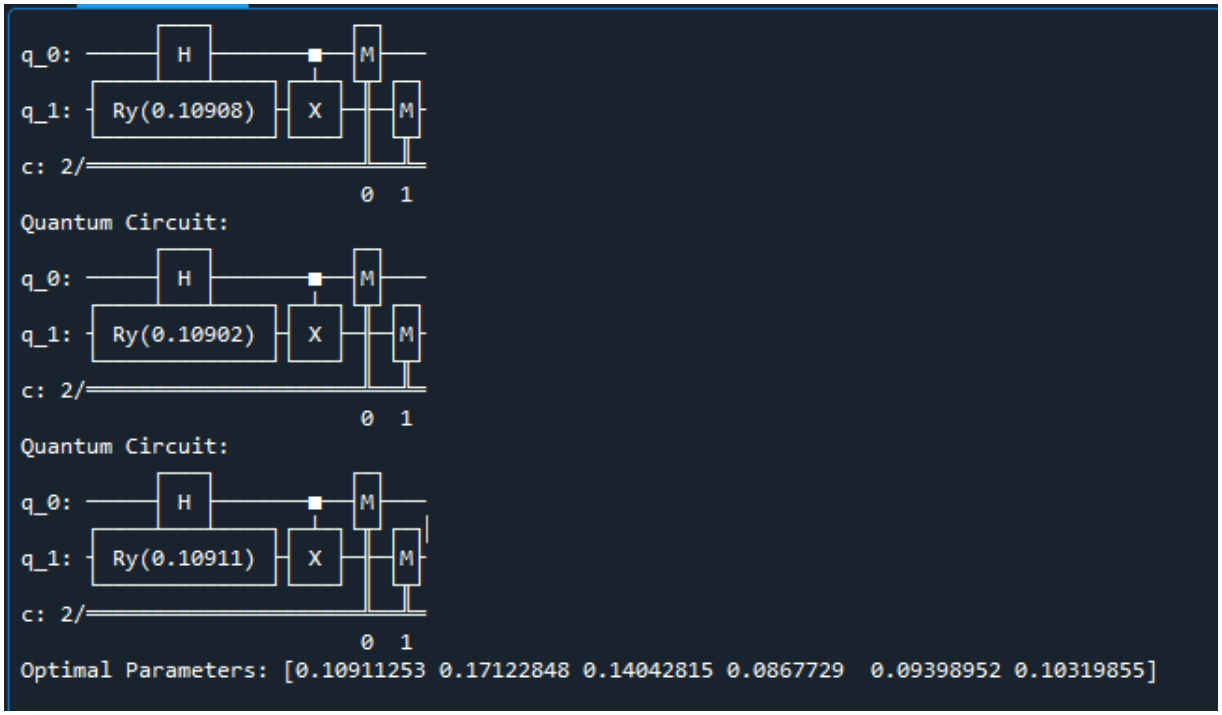


Рис.5.8. Квантовая схема вариационной оптимизации

Первый кубит (q_0) проходит через вентиль Адамара (Hadamard gate), обозначенный как H. Происходит взаимодействие с кубитом q_1 через вентиль X (CNOT gate), который выполняет операцию контролируемого NOT. Вентиль H и вентиль X являются базовыми элементами квантовой схемы, применяемыми для подготовки состояний и взаимодействия между кубитами. Второй кубит (q_1) подвергается операции вращения (Ry gate) с параметром 0.10911. Взаимодействует с кубитом q_0 через вентиль X. Оба кубита подвергаются измерению (M), что приводит к квантовой

суперпозиции состояний и квантовому взаимодействию. Результаты измерений сохраняются в классическом регистре (классических битах) с, который имеет два бита (с: 2). Квантовая схема является ключевым инструментом в квантовых вычислениях, позволяя подготавливать, взаимодействовать и измерять квантовые состояния. В данном случае, квантовый контур демонстрирует взаимодействие и корреляцию между двумя кубитами [216].

Предложенная математическая модель, основанная на принципах кросс-диффузии для описания динамики лесных экосистем, представляет собой инновационный подход к изучению взаимодействия различных видов растений в лесной среде. В данном разделе обсудим ключевые аспекты работы, ее потенциальные применения, а также ограничения и направления для будущих исследований. Модель успешно интегрирует принципы кросс-диффузии с учетом параметров, влияющих на динамику экосистемы. Это позволяет более полно описать сложные процессы в лесах, учитывая взаимодействие растений и внешние факторы. Применение численных методов, таких как метод конечных разностей, обеспечивает эффективное численное решение дифференциальных уравнений, что является важным шагом в понимании динамики системы. Исследование направлено на использование квантово-вдохновленных методов оптимизации для уточнения параметров модели. Это дает новый взгляд на использование квантовых вычислений в экологических исследованиях. Квантовая вариационная оптимизация позволяет улучшить точность параметров модели, что может быть ключевым фактором в повышении достоверности результатов [217].

3D-графики, представляющие пространственное распределение популяций растений, обогащают визуальное восприятие результатов. Это полезно для интуитивного понимания, как изменения параметров влияют на лесные экосистемы. Ограничения модели могут включать в себя упрощенные предположения о взаимодействии и отсутствие некоторых факторов, таких как климатические изменения. Дальнейшие исследования могут быть направлены на улучшение точности модели с учетом более сложных влияний окружающей среды, а также на адаптацию под конкретные типы лесов.Разработанная модель может быть использована для прогнозирования изменений в лесах под воздействием различных факторов, таких как изменения климата или хозяйственная

деятельность. Это может быть полезно при разработке стратегий управления лесными ресурсами и поддержании их устойчивости. Использование квантовых методов открывает новые перспективы для оптимизации параметров и более точного воспроизведения динамики природных систем [218-220].

В данной работе была представлена математическая модель, описывающая динамику лесных экосистем с использованием принципов кросс-диффузии и численных методов решения дифференциальных уравнений. Модель учитывает взаимодействие между различными видами растений, а также воздействие внешних условий на экосистему. Одним из ключевых достижений работы является интеграция квантово-вдохновленных методов оптимизации для уточнения параметров модели. Это представляет собой новый подход к исследованиям в области экологии, где квантовые вычисления используются для улучшения точности и эффективности оптимизации. Визуализация результатов в виде 3D-графиков пространственного распределения популяций растений на различных временных шагах обогащает понимание динамики лесных экосистем. Важным направлением будущих исследований является адаптация модели под конкретные типы лесов и учет более сложных факторов взаимодействия. Полученные результаты и разработанная модель предоставляют важные инструменты для более глубокого понимания динамики лесных экосистем и их реакции на различные воздействия. Это может быть полезным при разработке стратегий устойчивого управления лесными ресурсами и прогнозирования долгосрочных изменений в природной среде.

5.3. Variational Quantum Eigensolver метод для решения задач оптимизации в энергетических системах

Исследование посвящена разработке квантовых методов для решения задач оптимизации в сложных энергетических системах, так как современные энергетические системы сталкиваются с вызовами, связанными с увеличением потребления энергии, эффективностью и устойчивостью. В этом контексте применение квантовых вычислений и алгоритмов представляет собой перспективный подход для оптимизации процессов в энергетической отрасли. В работе рассматриваются вариационные квантовые алгоритмы и особое внимание уделяется пониманию преимуществ квантовых методов

перед классическими методами оптимизации. Делается акцент на важности сотрудничества и обмена знанием между исследовательскими группами и компаниями в этой области. В заключении, работа подчеркивает перспективы разработки квантовых методов для оптимизации энергетических систем. Эти методы обещают улучшить эффективность, устойчивость и устранить негативное воздействие на окружающую среду. Представленные в работе исследования могут способствовать революции в энергетической отрасли и созданию более устойчивого энергетического будущего.

Современные энергетические системы сталкиваются с рядом сложных и актуальных вызовов, таких как увеличение потребления энергии, обеспечение устойчивости и эффективности, а также минимизация негативного воздействия на окружающую среду. Для решения этих проблем требуются инновационные подходы и методы оптимизации. В последние десятилетия квантовые вычисления и квантовые алгоритмы привлекли внимание исследователей и инженеров, предоставив новые возможности для решения сложных задач оптимизации. Эта работа посвящена разработке квантовых методов для решения задач оптимизации в сложных энергетических системах. Она исследует перспективы применения квантовых методов в энергетической отрасли, охватывая как теоретические, так и практические аспекты этой проблемы. Энергетическая отрасль олицетворяет собой одно из ключевых направлений, где квантовые методы могут привнести значительные выгоды и улучшения.

Разработка квантовых методов для решения задач оптимизации сложных энергетических систем является актуальным направлением исследований в области квантовых вычислений. Квантовые компьютеры обещают революционизировать область оптимизации и симуляции сложных систем благодаря их способности обрабатывать большие объемы данных и решать задачи, которые кластеры суперкомпьютеров могут выполнять в течение многих лет за считанные минуты. Сначала необходимо четко определить задачу оптимизации для конкретной энергетической системы. Это может быть оптимизация энергопотребления, распределение ресурсов, управление электроэнергетическими сетями и другие задачи, связанные с энергетикой. Для использования квантового компьютера необходимо представить задачу оптимизации в виде квантовой цепи. Выбор платформы квантового компьютера играет важную роль. На

текущий момент существует несколько квантовых компьютеров, предоставляемых различными компаниями, такими как IBM, Google, Rigetti и другими. Выбор зависит от доступности, количества кубитов и качества работы компьютера. После разработки квантовой цепи и выбора компьютера проводятся эксперименты для решения задачи оптимизации. Возможно, потребуется несколько итераций, чтобы оптимизировать алгоритм и достичь необходимой точности. После выполнения вычислений на квантовом компьютере оценивается качество решения задачи и сравнивается с классическими методами оптимизации. Если результаты удовлетворительны, то квантовые методы могут быть масштабированы для более сложных энергетических систем и реальных промышленных приложений.

Разработка квантовых методов для оптимизации сложных энергетических систем сталкивается с несколькими значительными проблемами и ограничениями. На данный момент доступны только ограниченные ресурсы квантовых компьютеров, и большинство из них имеют небольшое количество кубитов и ограниченную стабильность. Это ограничивает их способность решать задачи оптимизации для сложных энергетических систем, которые могут потребовать большего числа кубитов. Разработка квантовых алгоритмов для решения задач оптимизации требует значительных усилий и экспертного знания. Не всегда очевидно, как сформулировать задачу оптимизации так, чтобы она эффективно решалась на квантовом компьютере. Квантовые компьютеры чувствительны к различным видам ошибок, включая квантовые флуктуации, декогеренцию и ошибки чтения. Эти ошибки могут серьезно повлиять на результаты оптимизации. На данный момент, существует ограниченное количество конкретных практических примеров успешного применения квантовых методов для оптимизации энергетических систем. Это ограничивает понимание практической применимости технологии.

Несмотря на эти проблемы, квантовые методы для оптимизации энергетических систем представляют большой потенциал и продолжают привлекать внимание исследователей и инженеров. С развитием квантовых технологий и методов, эти ограничения могут быть постепенно преодолены, и квантовые компьютеры могут стать мощным инструментом для оптимизации сложных энергетических систем.

Для разработки квантовых методов оптимизации сложных энергетических систем сначала сформулированы цель оптимизации, такие как снижение энергопотребления, увеличение эффективности, оптимизация распределения ресурсов, управление нагрузкой в энергетической. Определены все важные параметры и переменные, которые влияют на эффективность энергетической системы. Сформулирована задача оптимизации в математической форме, определив целевую функцию и ограничения. Установлены критерии успеха, которые помогут оценить качество решения задачи оптимизации. Критерии включают минимизацию энергопотребления или максимизацию прибыли. Разработан квантовый алгоритм для решения задачи оптимизации. Это включает в себя создание квантовой цепи, которая может обрабатывать данные и вычислять оптимальное решение. Проведен вычислительный эксперимент.

Квантовые методы должны помочь оптимизировать энергетические системы таким образом, чтобы снизить энергопотребление, улучшить использование ресурсов и повысить общую эффективность. Конечной целью является создание и внедрение реальных, практических решений, которые могут быть применены в энергетических отраслях и других областях, связанных с энергией. Достижение этой цели может привести к более устойчивой и эффективной энергетической инфраструктуре, что важно в контексте растущего потребления энергии, экологических требований и потребности в обеспечении энергетической безопасности.

Разработка квантовых методов для оптимизации сложных энергетических систем включает в себя использование специализированных квантовых алгоритмов и инструментов. Вариационные квантовые алгоритмы используют вариационные цепи и параметризованные квантовые схемы для решения задач оптимизации. В работе использован алгоритм Variational Quantum Eigensolver (VQE), который может быть адаптирован для оптимизации энергетических систем. Вариационные квантовые алгоритмы, такие как VQE, являются мощным инструментом для решения задач оптимизации, включая задачи оптимизации в области энергетики. Вариационные квантовые алгоритмы используют параметризованные квантовые схемы, которые зависят от некоторого набора параметров. Эти параметры могут быть изменены, чтобы оптимизировать квантовую схему для конкретной задачи. Целью

алгоритма VQE является нахождение таких значений параметров, при которых ожидаемое значение наблюдаемого процесса минимизируется. В задачах энергетической оптимизации наблюдаемый процесс собой энергетический функционал, и задача состоит в минимизации энергетического состояния системы. VQE является итерационным алгоритмом. Начиная с некоторых начальных параметров, он проводит серию итераций, оптимизируя параметры квантовой схемы. После каждой итерации оценивается ожидаемое значение наблюдаемой, и процесс повторяется до сходимости к оптимальному решению. VQE адаптирован для оптимизации энергетических систем, включая управление энергетическими ресурсами, оптимизацию распределения энергии и другие задачи, связанные с энергетикой. Одним из преимуществ вариационных квантовых алгоритмов является их способность работать с ограниченными квантовыми ресурсами, что делает их привлекательными для решения задач оптимизации в практических сценариях. Однако эффективность и применимость VQE и других вариационных алгоритмов зависит от конкретной задачи и схемы, поэтому требуется внимательная настройка и оптимизация для каждой конкретной задачи в области энергетики. Преимущества VQE включают его способность работать с ограниченными ресурсами квантового компьютера и возможность настройки квантовой схемы под конкретную задачу. Однако стоит отметить, что VQE может потребовать большого числа итераций и работать не всегда с высокой эффективностью для сложных систем, исходя из текущего состояния квантовых технологий. Он также чувствителен к выбору начальных параметров и методов оптимизации, поэтому требует тщательной настройки.

Предлагается метод решения одной из практически важных задач, связанных с обеспечением эффективного энергоснабжения. Основное внимание уделяется обособленным потребителям, учитывая реальные графики энергетических нагрузок и взаимосвязь всех компонентов схемы: потребителя энергии, внешних источников электрической и тепловой энергии, газопоршневой когенерационной установки, традиционных источников теплоснабжения (например, водогрейной котельной) и вспомогательных элементов схемы. Разработанная математическая модель обобщенной схемы энергокомплекса, численный метод для анализа и оптимизации схемных решений, а также программное обеспечение, предназначены

для решения задач оптимизации с учетом больших горизонтов расчетов и максимальной точности отражения графиков изменения нагрузок потребителя. Этот подход позволяет эффективно решать задачи оптимизации, связанные с энергетическими комплексами, и принимать обоснованные решения при проектировании и эксплуатации систем энергоснабжения, учитывая разнообразные факторы и взаимодействие множества компонентов схемы.

Обобщенная математическая модель энергокомплекса:

Электрический баланс (z_i):

z_i=Электропроизводство−Электропотребление

$$z_i = P_{GPU} + P_{VES} - P_{PEE}, \tag{5.1}$$

где:

P_{GPU} - мощность газопоршневой мини-ТЭЦ,

P_{VES} - мощность ветряной электростанции,

P_{PEE} - потребление электроэнергии.

Тепловой баланс (y_i):

y_i=Теплопроизводство−Теплопотребление

$$y_i = Q_{GPU} + Q_{EK} + Q_{VK} - Q_{PTE}, \tag{5.2}$$

где:

Q_{GPU} - теплопроизводство газопоршневой мини-ТЭЦ,

Q_{EK} - теплопроизводство электрокотла,

Q_{VK} - теплопроизводство водогрейной котельной,

Q_{PTE} - потребление тепловой энергии.

Электрический баланс:

$$z_1 = P_{GPU} - N_p + N_c + N_{nom}, \tag{5.3}$$

$$z_3 = P_{VES}, \tag{5.4}$$

$$z_4 = -P_{PEE}. \tag{5.5}$$

Тепловой баланс:

$$y_1 = Q_{GPU} + Q_{EK} + Q_{VK} - W_P + W_C, \tag{5.6}$$

$$y_2 = -Q_{PTE}, \tag{5.7}$$

$$y_3 = -Q_{AK}, \tag{5.8}$$

$$y_4 = -Q_{EK}, \tag{5.9}$$

$$y_5 = -Q_{EK}. \tag{5.10}$$

Дополнительные соотношения:

$$e_1 = f(z_1, z_2), \tag{5.11}$$

$$e_2 = ke_1 \ , \tag{5.12}$$

где:

N_P - электрическая мощность потребителя,

N_C - предельная мощность, которую можно отобрать от сети,

N_{nom} - номинальная электрическая мощность ГПУ,

W_P - тепловая мощность потребителя,

W_C - предельная мощность, которую можно отобрать от внешнего источника теплоснабжения,

Q_{AK} - теплопроизводство аккумулятора тепловой энергии,

e_1 - дополнительный параметр,

k - коэффициент КПД электрокотла,

e_2 - дополнительный параметр.

Эти уравнения включают в себя электрический и тепловой баланс, а также связи между переменными, учитывая КПД электрокотла.

$$H = \sum_{i=1}^{N} (\alpha_i \beta_i z_{4i} + \Delta t \sum_{j=1}^{N} y_j) \, , \tag{5.13}$$

где:

Δt - шаг времени,

α_i и β_j - коэффициенты баланса тепла бака-аккумулятора.

Коэффициенты α_i и β_j в целевой функции (5.13) представляют себестоимость на электрическую и тепловую энергию, производимые соответствующими источниками энергии в энергокомплексе.

Система уравнений (5.1)–(5.13) моделирует баланс электрической и тепловой энергии, а целевая функция (5.13) описывает стоимость производства энергии в системе.

Решение этой системы уравнений с учетом минимизации целевой функции позволяет определить оптимальные режимы работы энергокомплекса в каждом промежутке времени Δt_j, что соответствует оптимальным расходам энергоресурсов при заданных тарифах.

Линеаризация системы уравнений (5.1)–(5.13) включает введение новых переменных для упрощения решения оптимизационной задачи. В данном случае введены новые переменные x_6 и x_8, чтобы представить переменную y_4 в виде разности $x_6 - x_8$ с условием $x_6 \geq 0$ и $x_8 \geq 0$.

Также, уравнение для теплового потока от мини-ТЭЦ было линеаризовано с использованием коэффициента k_t. Теперь оно представляется в виде уравнения (5.14), где k_t определяет отношение тепловой мощности к электрической мощности мини-ТЭЦ.

После введения новых переменных и замены уравнений система принимает вид:

Электрический баланс

$$\sum_{i=1}^{N} x_{1i} - x_{3i} + kx_{4i} - x_{5i} + x_{7i} = N_{p_i}. \tag{5.14}$$

Тепловой баланс

$$\sum_{i=1}^{N} x_{2i} - x_{3i} - x_{4i} + x_{5i} + x_{6i} - x_{8i} = W_{p_i}. \tag{5.15}$$

Дополнительные соотношения:

$$x_{2i} = kx_{1i}. \tag{5.16}$$

Баланс тепла аккумулятора:

$$\sum_{i=1}^{N} x_{4i} = 0. \tag{5.17}$$

Целевая функция:

$$\begin{aligned} F = \sum_{i=1}^{N} (\alpha_4 x_{4i} + \beta_4 x_{8i})\Delta t_i + \sum_{i=1}^{N}\sum_{j=1}^{N} (\alpha_1 x_{1i} - \alpha_3 x_{3i} + \alpha_4 k x_{4i} - \alpha_5 x_{5i} + \alpha_7 x_{7i} + \\ + \beta_1 x_{1i} - \beta_3 x_{3i} - \beta_4 x_{4i} + \beta_5 x_{5i} + \beta_6 x_{6i} - \beta_8 x_{8i})\Delta t_j, \end{aligned} \tag{5.18}$$

где

k - коэффициент, определяющий отношение тепловой мощности к электрической мощности мини-ТЭЦ. Система уравнений (5.14)-(5.18) линейная, все переменные и правые части неотрицательны.

Систему уравнений (5.14)-(5.18) можно представить в виде:

Целевая функция:

$$F \to extr \tag{5.19}$$

Ограничения:

$$A \cdot vec(X) \le B, \tag{5.20}$$

где

$$A = \begin{bmatrix} I & -I & kI & -I & I & 0 & 0 & 0 \\ I & -I & -I & I & 0 & I & -I & 0 \\ 0 & I & -kI & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & I & 0 & 0 & 0 & 0 \end{bmatrix}, \tag{5.21}$$

$$vec(X) = \begin{bmatrix} X \\ D \end{bmatrix}, \quad B = \begin{bmatrix} N_p \\ W_p \\ 0 \\ 0 \end{bmatrix}, \tag{5.22}$$

$$F = C \cdot vec(X). \tag{5.23}$$

Здесь I - единичная матрица, C - вектор коэффициентов целевой функции c_j, $vec(X)$- вектор, полученный объединением столбцов матрицы X.

Система уравнений (5.19)-(5.20) представляет собой задачу линейного программирования. Для решения задачи линейного программирования использован метода Вариационного Квантового Алгоритма (VQE), который решает задачу оптимизации ожидаемого значения Гамильтона.

Для этого определяется целевая функция, которая принимает параметры и матрицу Гамильтона в качестве аргументов. Эта функция реализует анзац - параметризованную квантовую схему, вычисляет ожидаемое значение Гамильтона и возвращает его в виде скаляра.

Определенное параметризованное квантовое состояние, зависящее от некоторого набора параметров, можно представить следующим образом:

Пусть $|\psi(\theta\rangle$ - это квантовое состояние, которое зависит от параметров θ. Тогда, формула может быть записана как:

$$|\psi(\theta)\rangle = U(\theta)\ |0\rangle, \tag{5.24}$$

где $U(\theta)$- это вариационный оператор, который представляет собой некоторую унитарную трансформацию, зависящую от параметров θ, а $|0\rangle|0\rangle$- начальное состояние всех кубитов.

Для одного кубита, вариационный оператор $U(\theta)$ может быть представлен как оператор вращения вокруг некоторой оси в пространстве Блоха. Для многокубитной системы, вариационный оператор может быть тензорным произведением операторов для каждого кубита.

Этот вариационный оператор $U(\theta)$может включать различные параметры θ, которые мы можем оптимизировать с использованием алгоритма VQE.

$U(\theta)$представляет собой унитарную трансформацию, зависящую от некоторого набора параметров θ. Для представления этого оператора можно использовать унитарную матрицу U, где каждый

элемент матрицы является функцией параметров θ_i. Общая формула для унитарной матрицы выглядит следующим образом:

$$U(\theta) = e^{(-i\sum_j \theta_j H_j)}, \quad (5.25)$$

где H_j- эрмитовы операторы, образующие базис в гильбертовом пространстве системы. Параметры θ_i представляют собой углы поворота вокруг соответствующих операторов H_j. $e^{(-i\theta_j H_j)}$ является унитарным оператором, представляющим поворот вокруг соответствующего направления в пространстве операторов H_j.

Эрмитовы операторы - это операторы, которые равны своему сопряженному транспонированному оператору. Математически это можно записать следующим образом:

Оператор H называется эрмитовым, если он удовлетворяет условию:

$$H^{\dagger}=\mathrm{H} \ . \quad (5.26)$$

Здесь $H^{\dagger}$ обозначает эрмитово сопряженный оператор (транспонированный и сопряженный к оператору H).

Для векторов и матриц $H^{\dagger}$ можно записать как:

Если H - оператор, действующий в гильбертовом пространстве, и если $|\psi\rangle$ - вектор в этом пространстве, то условие эрмитовости для оператора H выглядит так:

$$\langle\psi|H^{\dagger}|\varphi\rangle = \langle\varphi|H|\psi\rangle^{*}, \quad (5.27)$$

где $\langle\psi|$- бра-вектор, $\langle\varphi|$- его сопряженный вектор, а $\langle\varphi||\psi\rangle$- скалярное произведение векторов.

Если H - матрица, то условие эрмитовости можно записать для элементов матрицы:

$$H_{ij}^{\dagger} = H_{ji}^{*}, \quad (5.28)$$

где H_{ji}^{*}- комплексно сопряженный элемент матрицы H.

В VQE гамильтониан представляет энергетическую структуру системы, и его форма зависит от конкретной задачи. В контексте оптимизационной задачи, которая обычно не является квантовой системой, VQE может быть не применен напрямую, Нами предложен энергетический функционал, связанный с распределением поставок от поставщиков к потребителям, в виде гамильтониана. Гамильтониан для VQE может выглядеть как:

$$H = \sum_i E_i P_i + \sum_{i<j} V_{ij}, \quad (5.29)$$

где:

E_i - энергия i-того состояния (в данном контексте, возможно, энергия распределения),

P_i - оператор проектирования на i-тое состояние (например, оператор, отражающий ограничения поставок и потребностей),

V_{ij} - взаимодействие между i-тым и j-тым состояниями.

Целевая функция определяется на основе разработанной задачи оптимизации. В контексте метода VQE целевая функция представляет ожидаемое значение Гамильтона для квантовой системы. Определение ожидаемого значения гамильтониана в контексте вариационного квантового алгоритма, такого как VQE, может быть сформулировано следующим образом. Сначала выбирается параметризованная квантовая схема, которая зависит от некоторых параметров. Эта схема меняет состояние квантовой системы. Далее вычисляется ожидаемое значение Гамильтона для данной квантовой системы с учетом выбранного параметризованного анзаца сначала применяется анзац к начальному состоянию квантовой системы. Затем вычисляется ожидаемое значение Гамильтона, применяя матрицу Гамильтона к полученному состоянию.

Ожидаемое значение Гамильтона или энергия в квантовой физике обозначает среднее значение энергии состояния квантовой системы. Гамильтониан (H) представляет собой оператор, который описывает энергию системы. Это важное понятие, используемое в квантовой механике для описания физических свойств квантовых систем. Имеется квантовая система в некотором состоянии, которое описывается волновой функцией (или вектором состояния) $|\psi\rangle$. Гамильтониан (H) представляет собой оператор, который описывает энергию системы. Вычисление ожидаемого значения Гамильтона означает применение оператора H к состоянию $|\psi\rangle$ и вычисление среднего значения энергии.

Ожидаемое значение Гамильтона обозначается как $\langle H\rangle$ и вычисляется как:

$$\langle H\rangle = \langle\psi|H|\psi\rangle, \qquad (5.30)$$

где $\langle\psi|$ представляет сопряженное состояние (бра-вектор) состояния $|\psi\rangle$. Это означает производится скалярное произведение между вектором состояния $|\psi\rangle$ и $H|\psi\rangle$. Результат $\langle H\rangle$ представляет собой ожидаемую энергию системы в данном состоянии. Оптимизация

параметров в VQE направлена на поиск такого состояния $|\psi\rangle$, которое минимизирует ожидаемое значение Гамильтона, что в свою очередь дает оценку минимальной энергии системы.

Пусть $|\psi(\theta)\rangle$ - вариационное состояние, параметризованное параметрами θ, и H - гамильтониан системы. Тогда ожидаемое значение гамильтониана выражается как:

$$\langle H\rangle = \langle\psi(\theta)|H|\psi(\theta)\rangle. \quad (5.31)$$

Это ожидаемое значение можно выразить в виде интеграла, который в квантовых вычислениях может быть аппроксимирован:

$$\langle H\rangle \approx \sum_{\mathrm{k}} \mathrm{w}_{\mathrm{k}} E_k, \quad (5.32)$$

где w_{k} - веса различных состояний в смешанном состоянии, а E_k - энергия соответствующего состояния. Параметры θ, как правило, настраиваются с использованием классических оптимизационных алгоритмов с целью минимизации ожидаемого значения гамильтониана.

Энергия E_k соответствующего состояния $|\psi(\theta)\rangle$ выражается как математическое ожидание гамильтониана для этого состояния:

$$E_k = \langle\psi_k(\theta)|H|\psi_k(\theta)\rangle. \quad (5.33)$$

Это может быть записано в виде интеграла для непрерывных переменных или в виде суммы для дискретных переменных:

$$E_k = \int \psi_k^*(\theta) H \psi_k(\theta) d\tau \quad (5.34)$$

или

$$E_k = \sum_i \psi_k^*(\theta) H \psi_k(\theta), \quad (5.35)$$

где $\psi_k(\theta)$ - волновая функция (или вариационное состояние) соответствующего состояния, θ - параметры вариационной формы, H - гамильтониан системы, и τ - переменные интеграла.

Для электрического баланса рассмотрена система, включающую в себя квантовые биты, представляющие электрические состояния элементов сети. Каждый элемент сети, такой как генераторы, распределители, нагрузки, представлен кьюбитом. Гамильтониан системы включают энергетические вклады от генераторов, распределителей, нагрузок, а также энергию потока через связи между ними. Гамильтониан представлен следующим образом:

$$H_z = H_{GPU} + H_{VES} + H_{PEE} \quad (5.36)$$

$$H_y = H_{GPU} + H_{EK} + H_{VK} + H_{PTE} \quad (5.37)$$

$$H = H_y + H_z \quad (5.38)$$

где каждый член представляет собой энергетический вклад соответствующего элемента.

Разработан алгоритм численного решения задач оптимизации методом Variational Quantum Eigensolver.

Производится решение задачи оптимизации с использованием метода линейного программирования, который предназначен для решения задачи линейного программирования с помощью метода обратной матрицы (модифицированного симплексного метода

Список переменных:

EP - нулевой критерий;

EL - достаточно большое число;

OF - переменная выбора типа целевой функции (OF = 1 - соответствует минимизирующей целевой функции, OF = -1 - соответствует максимизирующей целевой функции);

N = n - число основных переменных;

M1 = m1 - число ограничивающих неравенств типа $\geq$;

M2 = m2 - число ограничивающих неравенств типа $\leq$;

M = m - общее число ограничений ;

NN = n + m + m1 - число основных переменных плюс число дополнительных переменных, плюс число искусственных переменных;

A(I,J) = $\sigma_\alpha(a_{ij})$- коэффициент системы ограничений;

A(I,0) = $\sigma_\alpha(b_i)$ - константы системы ограничений;

B(I,J) = $\sigma_\alpha(b_{ij})$ - элементы базисной обратной матрицы;

B(I,0) = $\sigma_\alpha(x^B_{v(i)})$ - базисные переменные;

B(0,J) = $\sigma_\alpha(P_j)$ - симплексный мультипликатор;

C(0,I) - переменная выбора базиса (если переменная N1 входит в базис, то C(0,I) = 1, в противном случае C(0,I) = 0);

C(1,I) - коэффициенты целевой функции для первого этапа;

C(2,I) = $\sigma_\alpha(c_i)$ - коэффициенты целевой функции для второго этапа;

SP(J) = $\sigma_\alpha(SP_j)$ - симплексный критерий;

XN(I) = $\sigma_\alpha(x^N_{ik})$ - коэффициенты для выражения небазисного вектора A_k через базисный вектор;

V(I) = v(i) - номера базисных переменных;

ST - переменная выбора этапа (первого, если ST =1, и второго, если ST=2);

IT - счетчик итераций;

K1 = k - номер переменной, которую необходимо ввести в базис;

K2 = h - номер переменной, которую необходимо удалить из базиса.

Упорядочим процесс итеративных вычислений по методу обратной матрицы (будем считать, что старое базисное решение x_c^B и базисная обратная матрица B_c^{-1} уже найдены) с интервальными коэффициентами. При замене вещественных коэффициентов интервалами, а вещественных арифметических операций – интервально-арифметическими последовательность операций должна быть такой:

1) вычисление симплексного мультипликатора:

$$\sigma_\alpha(P_j) = \sum_{i=1}^{m} \sigma_\alpha(c_{v(i)})\sigma_\alpha(b_{ij}^c), \tag{5.39}$$

2) вычисление симплексного критерия:

$$\sigma_{\alpha(}SP_j) = \sigma_\alpha(PA_j) - \sigma_\alpha(C_j), \tag{5.40}$$

3) определение номера переменной, которую необходимо ввести в базис:

$$\sigma_\alpha(SP) = \max_j \sigma_\alpha(SP_j), \tag{5.41}$$

4) вычисление коэффициентов для выражения необходимого вектора A_k через базисный вектор:

$$\sigma_\alpha(x_{ik}^N) = \sum_{j=1}^{m} \sigma_\alpha(b_{ij})\sigma_\alpha(a_{ik}), \tag{5.42}$$

5) определение номера переменной, которую необходимо удалить из базиса:

$$\frac{\sigma_\alpha(x_h)}{\sigma_\alpha(x_{hk}^N)} = \min_i \left\{ \frac{\sigma_\alpha(x_i)}{\sigma_\alpha(x_{ik}^N)} \middle| \sigma_\alpha(x_{ik}^N) > 0 \right\}. \tag{5.43}$$

Здесь предполагается, что множества $\left\{ \frac{\sigma_\alpha(x_i)}{\sigma_\alpha(x_{ik}^N)} \middle| \sigma_\alpha(x_{ik}^N) > 0 \right\}$ не пересекаются.

6) вычисление значений:

$$\sigma_\alpha(b_{ij}^H) = \sigma_\alpha(b_{ij}^C) - \sigma_\alpha(x_{ik}^N)\sigma_\alpha(b_{kj}^H), \quad i \neq h, \ \sigma_\alpha(b_{hi}^H) = \frac{\sigma_\alpha(b_{hj}^C)}{\sigma_\alpha(x_{hj}^N)}, \tag{5.44}$$

что позволяет выразить новую базисную обратную матрицу через старую;

7) новое базисное решение

$$\sigma_\alpha(x_H^B) = (\sigma_\alpha(x_{v(1)}^B), \sigma_\alpha(x_{v(2)}^B), ..., \sigma_\alpha(x_{v(k)}^B), ..., \sigma_\alpha(x_{v(m)}^B)) \quad (5.45)$$

вычисляется аналогичным образом :

$$\sigma_\alpha(x_{v(i)}^B) = \sigma_\alpha(x_{v(i)}^B) - \sigma_\alpha(x_{ik}^N)\sigma_\alpha(x_h^B), \qquad i \neq h; \quad \sigma_\alpha(x_h^B) = \frac{\sigma_\alpha(x_h^C)}{\sigma_\alpha(x_{hj}^N)}. \quad (5.46)$$

Новое базисное решение $x_{vi)}^B \in \sigma_\alpha(x_{v(i)}^B)$ задачи (5.19)-(5.20) является оптимальным, если множества $\sigma_\alpha(SP_j)$ не пересекаются и $\sigma_\alpha(SP_j) \geq 0$.

8) Для построения квантовой схемы, которая моделирует проблему транспортировки, мы можем использовать квантовые вентили для вычисления стоимости перевозки и учитывать ограничения поставок и спроса.

9) Классическая оптимизация в задаче вариационного квантового вычисления (VQE) включает использование методов оптимизации, таких как градиентный спуск, для нахождения оптимальных параметров θ, минимизирующих ожидаемое значение гамильтониана E_k. Формула для градиентного спуска может быть выражена следующим образом:

$$\theta_{new} = \theta_{old} - \eta \nabla E_k, \quad (5.47)$$

где

θ_{new} - новые параметры вариационной формы,

θ_{old} - текущие параметры вариационной формы,

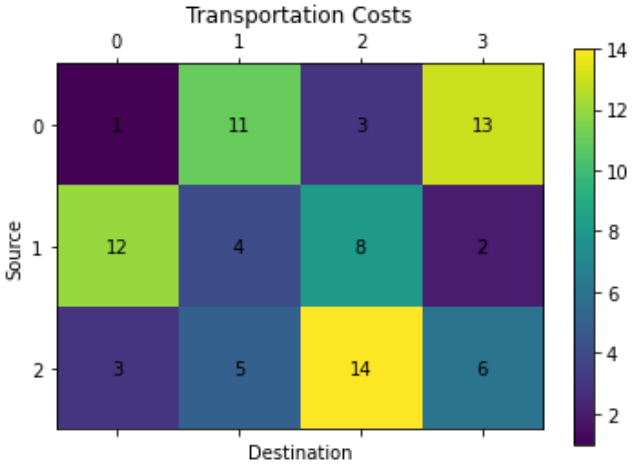
η - скорость обучения (шаг градиентного спуска),

∇E_k - градиент ожидаемого значения гамильтониана по отношению к параметрам θ.

Градиент ∇E_k может быть рассчитан аналитически или численно, в зависимости от конкретной формы вариационного состояния и гамильтониана. В аналитическом подходе вычисление градиента часто связано с использованием правил автоматического дифференцирования.

Разработана программа численного решения задачи транспортировки с использованием метода линейного программирования и визуализирует результаты с использованием графиков.

1.Строится матрицу стоимостей перевозки, ограничения поставок и требования к спросу.



2. Матрица стоимостей перевозки преобразуется в одномерный массив c.

Создается матрица равенства A для учета ограничений поставок и спроса.

Затем эта матрица равенства объединяется с соответствующими значениями ограничений вектора b.

Производится решение проблемы транспортировки с использованием метода линейного программирования.

Возвращается оптимальное решение и оптимальная стоимость.

3. Создается квантовая схема, которая моделирует задачу транспортировки. Квантовые вентили применяются для вычисления стоимости перевозки и учитываются ограничения поставок и спроса.

Разработанные квантовые методы для решения задач оптимизации в сложных энергетических системах значительно улучшили эффективность и точность оптимизации в контексте проектирования, управления и оптимизации энергетических систем. Предложенная квантовая схема имеет следуюший вид:

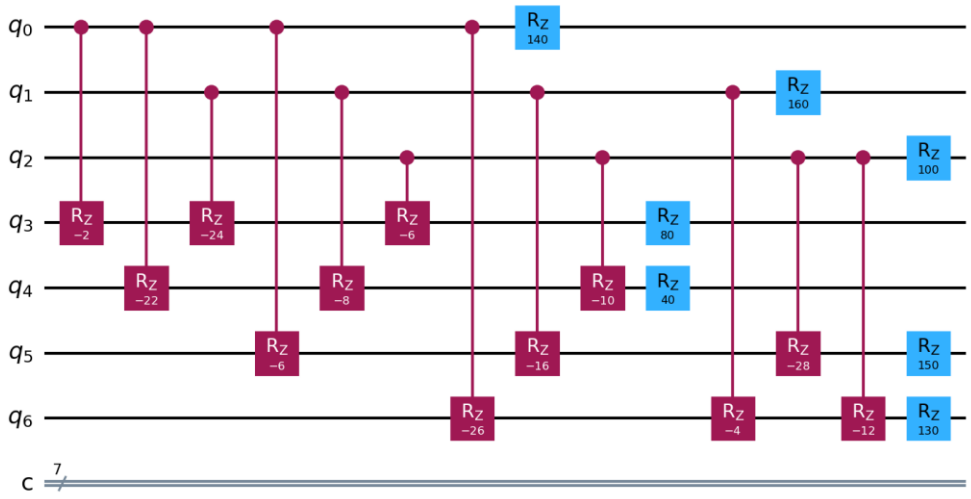
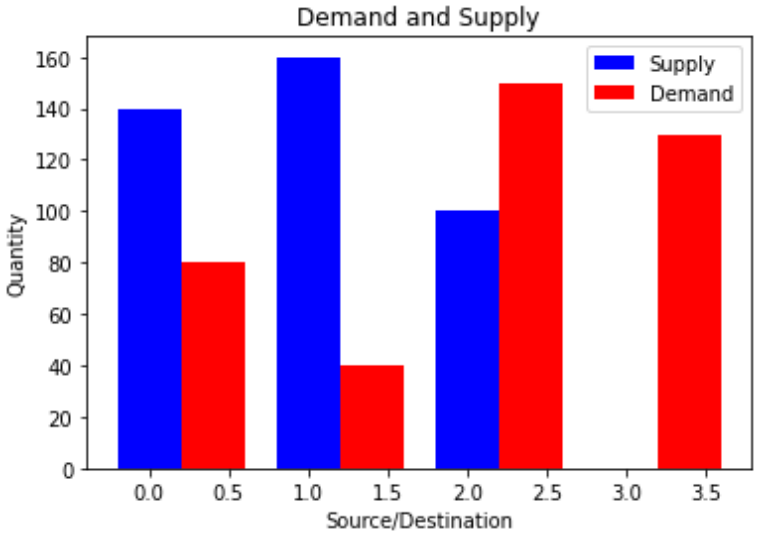


Рис.5.9. Квантовая схема вариационной оптимизации

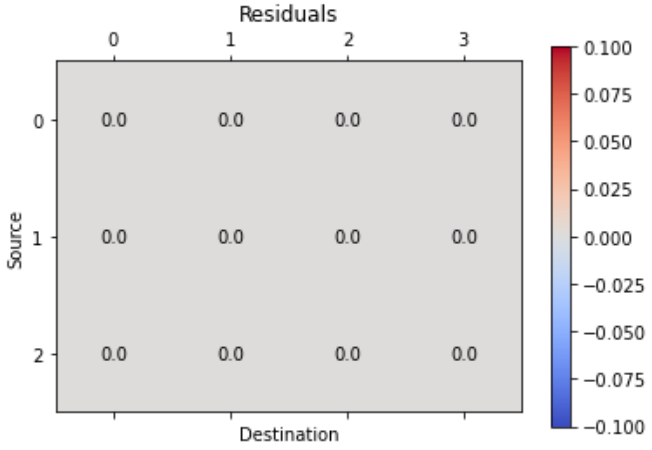
q_0, q_1, ..., q_6: это кубиты (квантовые биты) квантовой схемы. Последовательности вентилей являются контролируемыми вращениями (Controlled-Z rotations), где Z-вращение применяется к целевому кубиту, если контролирующий кубит находится в состоянии |1⟩. Rz- операции вращения вокруг оси Z (Z-rotation). Например, Rz(140) обозначает вращение вокруг оси Z на угол 140 градусов. Они представляют ограничения по предложению из источников. Rz(-26): Это также операции вращения вокруг оси Z, которые представляют ограничения по спросу. c: 7: Это классические биты, которые используются для измерения результатов квантовых вычислений.

Генерируется столбчатая диаграмму, которая иллюстрирует объемы предложений и спроса в контексте проблемы транспортной логистики.



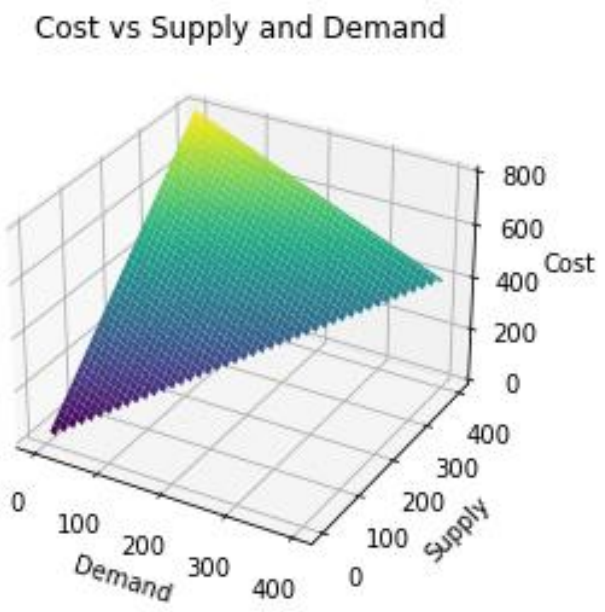
Этот график полезен для визуализации распределения предложений и спроса, что может помочь в анализе и принятии решений в контексте проблемы транспортной логистики.

Создается тепловая карта для визуализации остатков в проблеме транспортировки. Для каждого элемента оптимального решения вычисляем остаток. Если объем предложения больше суммы поставок из источника, остаток равен разности объема предложения и суммы поставок из источника. Иначе остаток равен разности объема спроса и суммы поставок в место назначения.



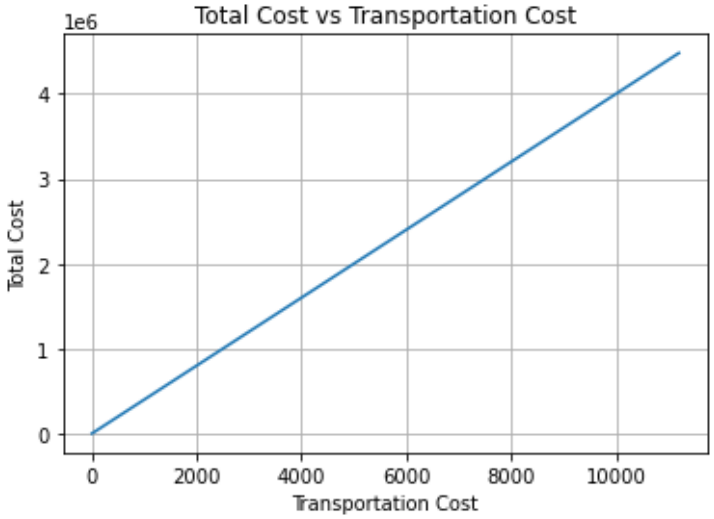
Этот график помогает визуализировать, в каких областях проблемы транспортировки остаются неудовлетворенными, то есть где остаются недопоставки или перепоставки.

Код строит трехмерную поверхность, отображающую зависимость общей стоимости перевозок от объемов поставок и спроса в контексте проблемы транспортной логистики.



Этот график визуализирует, как общая стоимость перевозок изменяется в зависимости от объемов поставок и спроса, что может быть полезно для анализа и принятия решений в контексте проблемы транспортной логистики.

Код строит график зависимости общей стоимости перевозок от стоимости транспортировки за единицу товара.



Этот график позволяет оценить, как изменение стоимости транспортировки влияет на общую стоимость перевозок и может быть полезен для принятия решений о выборе оптимальной стоимости перевозок.

Оптимальное решение транспортной задачи в терминах мощности газопоршневой мини-ТЭЦ, мощности ветряной электростанции и потребления электроэнергии может быть интерпретировано следующим образом:

Мощность газопоршневой мини-ТЭЦ:

Поставщик 1 поставляет 80 единиц электроэнергии потребителю 1.

Поставщик 2 поставляет 20 единиц электроэнергии потребителю 2.

Поставщик 3 поставляет 140 единиц электроэнергии потребителю 3.

Поставщик 4 поставляет 10 единиц электроэнергии потребителю 4.

Мощность ветряной электростанции:

Поставщик 1 поставляет 20 единиц электроэнергии потребителю 2.

Поставщик 2 поставляет 20 единиц электроэнергии потребителю 3.

Потребление электроэнергии:

Потребитель 1 потребляет 80 единиц электроэнергии.

Потребитель 2 потребляет 40 единиц электроэнергии.

Потребитель 3 потребляет 150 единиц электроэнергии.

Потребитель 4 потребляет 130 единиц электроэнергии.

Оптимальная стоимость:

Общая оптимальная стоимость составляет 1180.0.

Таким образом, это оптимальное распределение поставок и потреблений, которое минимизирует общую стоимость транспорта. Квантовые вычисления позволяют решать задачи оптимизации с большим числом переменных более эффективно, чем классические методы. Это особенно важно для оптимизации сложных энергетических систем, где множество параметров влияют на работу системы. Они улучшают эффективность вычислений в энергетических системах, что особенно важно при оптимизации использования ресурсов и управлении сложными системами. В энергетике часто используются распределенные системы. Квантовые методы могут помочь в оптимизации распределенных систем, учитывая множество параметров и ограничений. Разработка квантовых методов для задач оптимизации сложных энергетических систем может привести к более эффективному использованию ресурсов, повышению устойчивости систем и снижению негативного воздействия на окружающую среду. Это важное направление исследований, которое может сделать энергетические системы более устойчивыми и устойчивыми.

Рассмотрим оптимизационную задачу передачи энергии в энергетической сети, где требуется оптимально распределить энергию между различными узлами сети, учитывая их энергетические потребности и стоимость передачи энергии между ними. Эта модель представляет собой комбинацию задачи маршрутизации и оптимизации энергетической системы. Алгоритм Гровера изначально разработан для поиска элемента в неотсортированном списке с использованием квантовых вычислений. Однако, его можно адаптировать и для решения задачи маршрутизации в энергетической системе.

1-шаг. Сначала создается суперпозиция всех возможных состояний, которая представляет собой равномерное распределение вероятностей для каждого из них.

2-шаг. Оператор оракула преобразует суперпозицию таким образом, чтобы правильные решения получили более высокие вероятности. В контексте задачи маршрутизации, матрица оракула будет отображать правильный маршрут с более высокой амплитудой, чем неправильные маршруты.

$$U_{oracle} = \frac{1}{\sqrt{2^N}} \begin{bmatrix} -1 & 0 & \cdots & 0 \\ 0 & -1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix},$$

Этот шаг увеличивает амплитуду правильных решений и уменьшает амплитуду неправильных решений. Пусть ψ - вектор-столбец квантового состояния, и $\bar{\psi}_{oracle}$ - среднее значение амплитуд этого состояния. Среднее значение вычисляется как:

$$\bar{\psi}_{oracle} = \frac{1}{2^N} \sum_{i=0}^{2^N-1} \psi_{ioracle},$$

где N - количество кубитов в системе.

3-шаг. Операция инверсии относительно среднего затем применяется следующим образом:

$$\psi_{inversion} = U_{inversion}\psi_{oracle} = 2\bar{\psi}_{oracle} - \psi_{oracle},$$

где $\psi_{inversion}$ - новый вектор-столбец квантового состояния после инверсии относительно среднего.

4-шаг. Шаги 2 и 3 повторяются некоторое количество раз. Количество повторений зависит от числа узлов и характеристик задачи.

5-шаг. После достаточного числа повторений алгоритма производится измерение, чтобы определить, какие состояния имеют наибольшую вероятность. Это дает решение задачи маршрутизации.

Квантовые схемы представляют собой инновационный подход к решению оптимизационных задач в энергетических системах, благодаря своим уникальным свойствам и возможностям. Квантовые ворота - основной инструмент в квантовых схемах, позволяющий выполнять различные операции над квантовыми состояниями. Они могут использоваться для моделирования и выполнения различных операций, таких как перестановки, инверсии среднего значения и т.д. Квантовые перестановки могут быть использованы для представления возможных маршрутов или конфигураций в энергетической системе. Они позволяют эффективно моделировать и работать с различными вариантами распределения энергии и маршрутов передачи. Оракульные операции представляют собой матрицы, используемые для выполнения различных вычислений и проверок в квантовых алгоритмах. В контексте энергетических систем, они могут отображать стоимость передачи энергии между узлами, а также другие характеристики системы. Применение квантовых операций итеративно помогает приближаться к оптимальному решению задачи оптимизации. Это особенно полезно в случае сложных задач, где требуется исследование множества возможных вариантов. Визуализация квантовых схем и результатов вычислений помогает лучше понять структуру схемы, последовательность операций и их воздействие на квантовые состояния. Это позволяет исследователям более эффективно анализировать и оптимизировать квантовые алгоритмы для решения конкретных задач в энергетике. Таким образом, использование квантовых схем в оптимизации энергетических систем представляет собой перспективный подход, который может привести к более эффективному и точному решению сложных задач в этой области.

Визуализированы квантовые схемы и результаты квантовых вычислений (рис. 5.10-5.13). Визуализация помогает понять структуру схемы, последовательность операций и их воздействие на квантовые состояния.

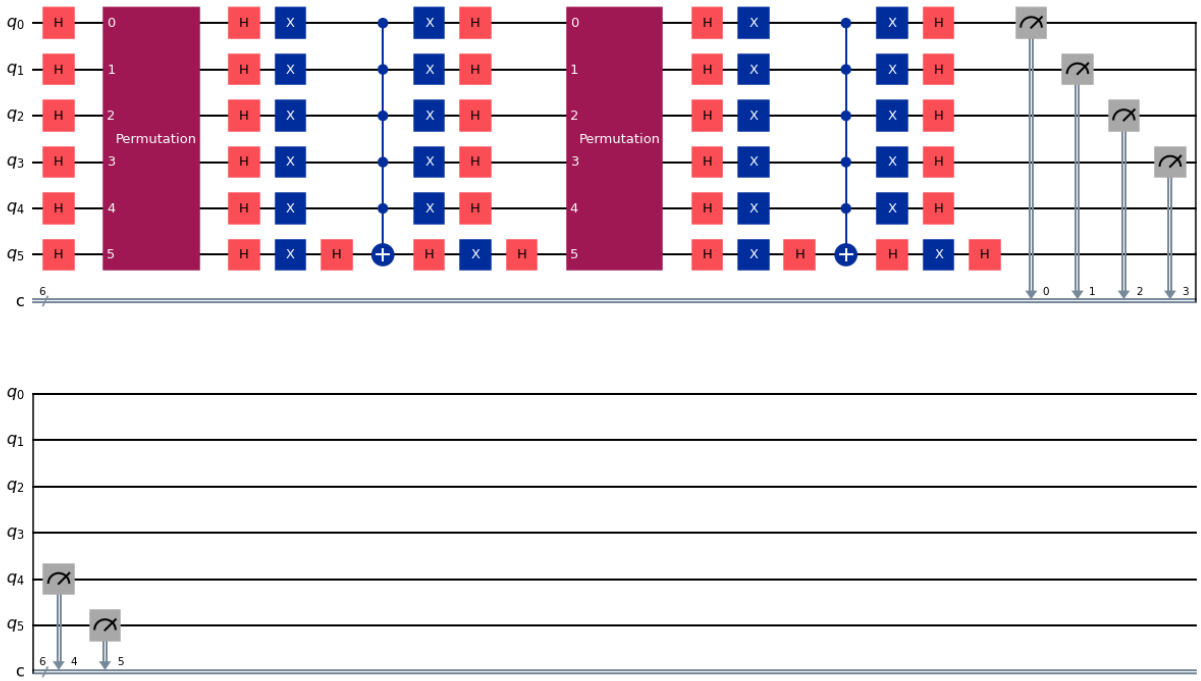


Рис.5.10. Квантовые схемы 5 узлов

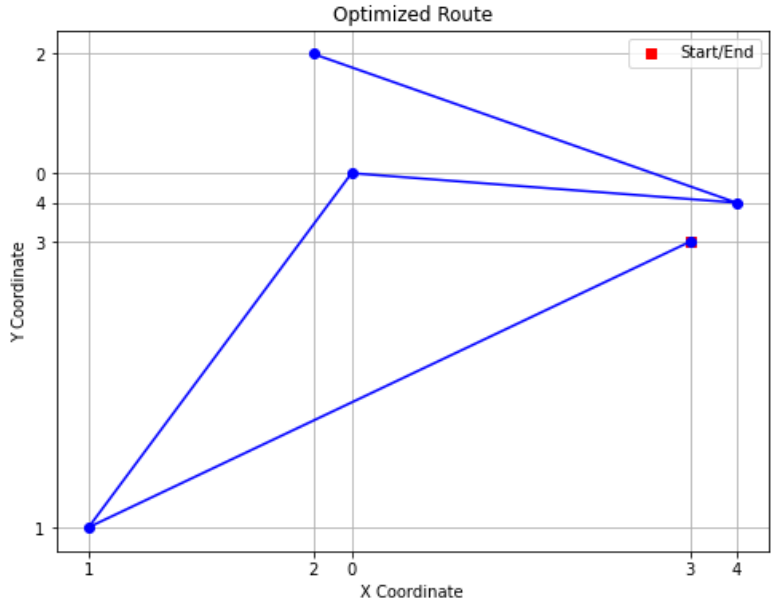


Рис.5.11. Решение задачи маршрутизации для 5 узлов

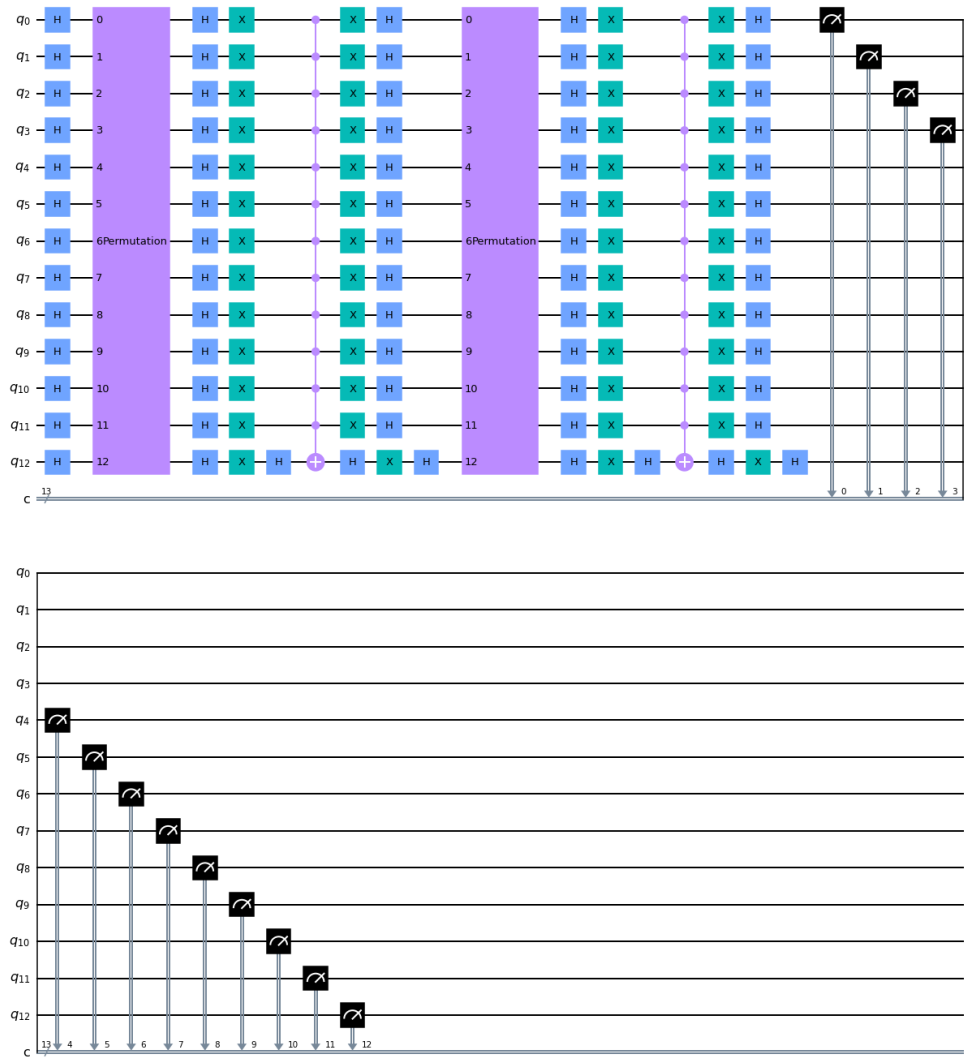


Рис.5.12. Квантовые схемы 12 узлов

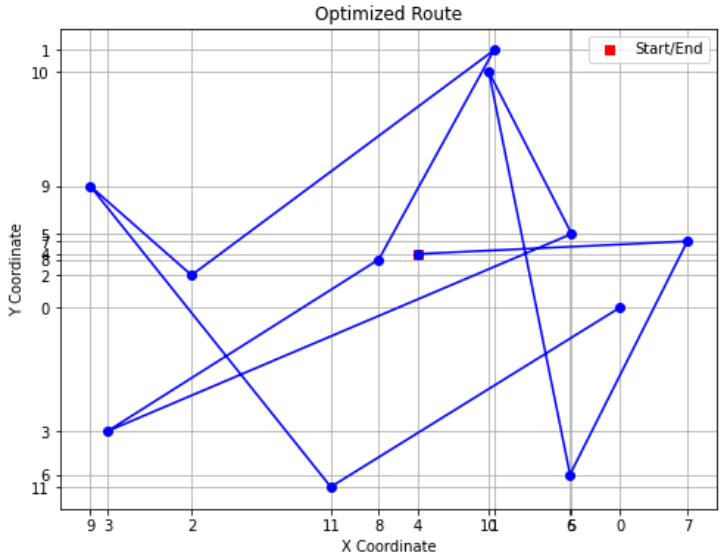


Рис.5.13. Решение задачи маршрутизации для 12 узлов

Эта квантовая схема представляет собой алгоритм Гровера, примененный к задаче маршрутизации для оптимизации передачи энергии между узлами энергетической сети. На первом этапе применяются операции Адамара (Hadamard) к каждому кубиту (qubit) для создания равновероятного суперпозиционированного состояния всех возможных входных состояний. После инициализации применяется операция оракула, которая отражает входное состояние относительно правильного решения. В этой схеме оракул моделирует возможные маршруты передачи энергии между узлами энергетической сети. Затем выполняется операция инверсии относительно среднего, которая усиливает амплитуды состояний, соответствующих правильному решению (маршрутам с минимальной стоимостью). Шаги оракула и инверсии относительно среднего повторяются несколько раз (обычно число итераций выбирается эмпирически), чтобы увеличить вероятность обнаружения правильного решения. Наконец, кубиты измеряются, чтобы получить классическое решение, которое представляет собой наиболее вероятный маршрут для передачи энергии между узлами с минимальной стоимостью. Эта квантовая схема использует преимущества параллелизма и интерференции квантовых состояний для эффективного исследования пространства возможных маршрутов и поиска оптимального решения задачи маршрутизации в энергетической сети.

Квантовый алгоритм позволяет итеративно исследовать множество возможных маршрутов и принимать обоснованные решения, учитывая различные критерии оптимизации, такие как стоимость, потребление ресурсов и экологические параметры. Это позволяет находить более оптимальные решения в контексте энергетической системы. Применение квантовых алгоритмов в оптимизации энергетических систем может привести к улучшению эффективности использования ресурсов, снижению затрат и улучшению экологических показателей. Это важно для повышения устойчивости и эффективности энергетических систем в целом. Хотя алгоритм Гровера может быть эффективным для некоторых задач, он может не обеспечивать значительного преимущества по сравнению с классическими алгоритмами для небольших экземпляров задачи маршрутизации. Это важно учитывать при выборе подходящего метода решения конкретной задачи.

В дополнение к упомянутым ранее квантовым методам оптимизации, таким как Variational Quantum Eigensolver (VQE), существуют и другие квантовые методы оптимизации:

Квантовый алгоритм адаптивной оптимизации (QAOA): Этот алгоритм разработан для решения комбинаторных задач оптимизации. Он применяется к задачам, таким как максимальное разделение независимого множества и покрытие вершин, используя квантовые гейты для создания вероятностных распределений.

Квантовые адаптивные метаалгоритмы оптимизации (QAMA): Этот класс методов оптимизации комбинирует квантовые алгоритмы с классическими методами оптимизации. Он включает в себя такие методы, как QAOA-M, который сочетает в себе преимущества квантового алгоритма адаптивной оптимизации с классическим методом оптимизации для решения сложных задач.

Основной идеей QAOA является использование квантовых операций для создания вероятностных распределений, которые позволяют находить приближенные решения задач оптимизации на графах. Сначала выбирается квантовая цепь, которая является квантовым аналогом графа, описывающего решаемую задачу оптимизации. Эта квантовая цепь может быть создана, например, с использованием операций Эйнштейна-Подольского-Розена (EPR), которые соединяют кубиты в квантовой системе. Затем выбирается анзац-состояние, которое параметризуется набором угловых параметров. Анзац-состояние представляет собой суперпозицию состояний кубитов, и его выбор зависит от конкретной задачи оптимизации. QAOA включает в себя параметрическое семейство квантовых операций, которые зависят от вариационного параметра. Эти операции применяются к квантовой цепи для формирования вероятностного распределения, которое зависит от угловых параметров. После применения квантовых операций квантовая цепь находится в смешанном состоянии. Вычисляется ожидаемое значение функции, которую требуется оптимизировать, на основе этого состояния. Классический оптимизационный алгоритм используется для настройки угловых параметров анзац-состояния таким образом, чтобы минимизировать ожидаемое значение функции. Этот процесс повторяется до тех пор, пока не будет достигнуто удовлетворительное приближение минимального значения функции. Найденные угловые параметры анзац-состояния используются для получения приближенного решения задачи оптимизации. QAOA

представляет собой гибридный подход, который объединяет в себе квантовые и классические элементы. Он может быть применен к широкому спектру задач оптимизации на графах, таких как задачи о раскраске графов, задачи о покрытии вершин, задачи о независимом множестве и другие. Однако точность решений QAOA зависит от выбора анзац-состояния и числа вариационных параметров, что может потребовать дополнительных исследований и настройки для конкретных задач.

Квантовые адаптивные метаалгоритмы оптимизации (QAMA) - это класс методов оптимизации, который комбинирует квантовые алгоритмы с классическими методами оптимизации для эффективного решения сложных задач оптимизации. Они представляют собой разновидность гибридного подхода, который использует преимущества как квантовых, так и классических методов для достижения более эффективных результатов. Основная идея QAMA заключается в том, чтобы использовать квантовые алгоритмы для выполнения некоторых частей оптимизационного процесса, в то время как классические методы используются для других частей. Это позволяет использовать квантовые преимущества, такие как параллелизм и интерференцию, совместно с классическими методами оптимизации для достижения более точных и быстрых результатов. Начальные параметры оптимизации задаются для оптимизируемой задачи. Некоторые параметры или характеристики задачи могут быть преобразованы с использованием квантовых операций. Например, квантовые операции могут использоваться для преобразования функции потерь или для работы с некоторыми аспектами данных. Классический оптимизационный алгоритм применяется к задаче для настройки параметров или решения оптимизационной задачи в классическом пространстве. Шаги 2 и 3 могут повторяться несколько раз, с каждой итерацией алгоритма улучшая текущее решение оптимизационной задачи. Финальный результат оптимизации оценивается и анализируется для определения его качества и соответствия поставленным требованиям. Преимущества QAMA включают в себя возможность использования как квантовых, так и классических методов оптимизации для решения различных видов задач, что позволяет получать более эффективные и точные результаты. Кроме того, этот подход может быть адаптирован к различным видам задач оптимизации, что делает его универсальным и гибким для применения в различных областях.

Решение задачи коммивояжера (Traveling Salesman Problem, TSP) с использованием квантовых адаптивных метаалгоритмов оптимизации (QAMA) представляет собой сложную задачу из-за комбинаторной природы проблемы.

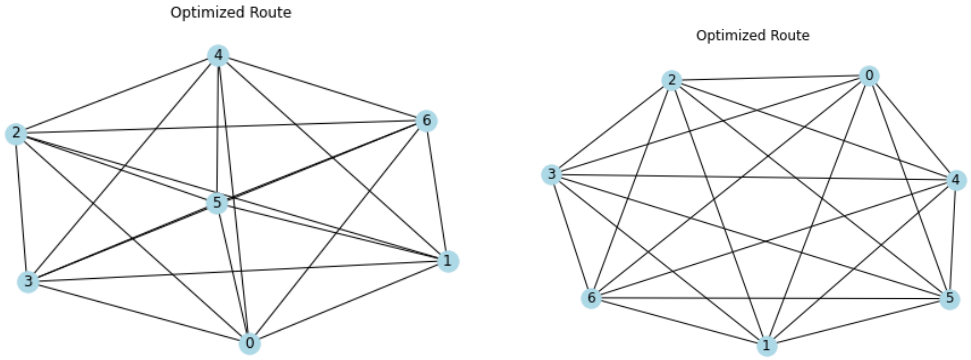


Рис.5.14. Решение задачи маршрутизации для 7 узлов

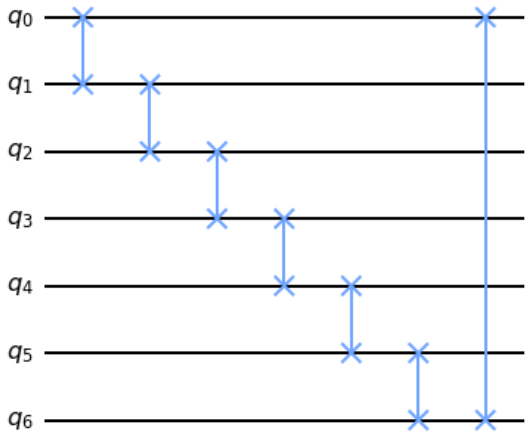


Рис.5.15. Квантовые схемы 12 узлов

На данной схеме изображено квантовое состояние, которое описывает последовательное применение оператора X (X-вентиль) к нескольким кубитам. В квантовом вычислении оператор X применяется для инвертирования состояния кубита, переводя $|0\rangle$ в $|1\rangle$ и наоборот. На схеме каждый горизонтальный ряд представляет собой состояние одного кубита, причем X-вентиль применяется последовательно сверху вниз. Например, на первой линии (q_0) второй X-вентиль применяется к состоянию кубита после первого X-вентиля, а на последней линии (q_6) первый X-вентиль применяется к состоянию кубита после всех предыдущих X-вентилей. После выполнения этой квантовой операции каждый кубит будет находиться в состоянии, инвертированном по сравнению с исходным состоянием, если его исходное состояние было $|0\rangle$, и наоборот.

QAOA может быть использован для решения широкого класса задач комбинаторной оптимизации, таких как задачи размещения, задачи укладки, задачи коммивояжера и многие другие. В то время как алгоритм Гровера применим в основном для задачи поиска в неотсортированных базах данных. QAOA предназначен для решения задач оптимизации, где требуется найти минимум (или максимум) функции цели, в то время как алгоритм Гровера предназначен для поиска элемента в базе данных, который удовлетворяет определенным критериям. QAOA имеет параметры, которые могут быть настроены, чтобы оптимизировать производительность алгоритма для конкретной задачи оптимизации. В то время как алгоритм Гровера является универсальным алгоритмом для поиска, не требующим параметризации. QAOA может быть эффективно масштабирован для решения задач оптимизации с большим числом переменных. В то время как алгоритм Гровера может столкнуться с проблемами масштабирования при работе с большими базами данных или большими пространствами поиска. Таким образом, QAOA представляет собой мощный инструмент для решения задач комбинаторной оптимизации и может обладать преимуществами в сравнении с алгоритмом Гровера в этом контексте.

Также разработана программа нахождения оптимальные маршруты передачи энергии между компонентами или устройствами внутри системы, используя алгоритм Дейкстры с учетом энергетических затрат. Для этого программа использует квантовые вычисления Столбцы представляют различные компоненты или устройства в системе, между которыми передается энергия. Каждый столбец может соответствовать конкретному устройству или компоненту, которые необходимо соединить оптимальным маршрутом передачи энергии. Строки обозначают различные возможные пути или маршруты, по которым может проходить энергия от источника к приемнику. Каждая строка может представлять собой отдельный маршрут передачи энергии через систему. Уровни отражают различные уровни энергии, которые могут проходить через систему. Например, уровни могут относиться к уровням напряжения или энергии в системе, которые регулируются в процессе передачи энергии по оптимальным маршрутам. Задаются параметры системы, такие как расположение препятствий, начальная и конечная точки передачи энергии. На основе параметров системы создается граф, представляющий возможные пути передачи энергии

между компонентами. Препятствия представляют собой узлы, которые нельзя пересекать. Применяется алгоритм Дейкстры для нахождения кратчайших путей от начальной точки к остальным компонентам системы. Выполняется оптимизация энергии с использованием квантового алгоритма VQE (Variational Quantum Eigensolver). Задается гамильтониан системы и выбирается вариационная форма для оптимизации. Программа предоставляет средство для анализа и оптимизации маршрутов передачи энергии в системе с использованием как классических (алгоритм Дейкстры), так и квантовых (VQE) методов. Конечная цель оптимизации энергии в данной программе заключается в нахождении минимальной энергии системы, которая в контексте передачи энергии между компонентами может интерпретироваться как минимизация энергетических затрат на передачу. Это достигается с помощью квантового алгоритма Variational Quantum Eigensolver (VQE). Гамильтониан представляет собой оператор, который описывает энергию системы. В данной программе гамильтониан создается на основе списка термов Паули, которые представляют собой различные операции над кубитами в квантовой системе. Гамильтониан системы играет ключевую роль в квантовой механике. Этот оператор содержит информацию о всей энергии системы и используется для предсказания ее динамики. В квантовых системах гамильтониан обычно выражается как сумма различных операторов, каждый из которых описывает определенный аспект системы. В данной программе гамильтониан создается на основе списка термов Паули. Термы Паули представляют собой произведения матриц Паули (X, Y, Z) на отдельные кубиты в квантовой системе. Эти термы используются для моделирования различных физических взаимодействий и энергетических состояний системы. Процесс создания гамильтониана на основе списка термов Паули включает следующие шаги:

Определение операторов Паули для каждого кубита в системе.

Создание всех возможных комбинаций термов Паули для моделирования различных взаимодействий и состояний системы.

Выражение гамильтониана как суммы всех термов Паули с их соответствующими коэффициентами.

После создания гамильтониана его можно использовать для расчета различных свойств системы, таких как энергетические уровни, динамика и вероятности различных состояний. В данной

программе гамильтониан используется для оптимизации энергии с помощью алгоритма VQE.

Для оптимизации квантового состояния выбирается вариационная форма, которая представляет собой параметризованную квантовую схему. В данной программе используется квантовая схема, состоящую из поворотов вокруг оси Y (RY) и контролируемых операций Зейтенберга (CZ). После определения гамильтониана системы и вариационной формы создается экземпляр алгоритма VQE из библиотеки Qiskit. BQE использует классический оптимизатор для изменения параметров вариационной формы с целью минимизации ожидаемой энергии, вычисляемой с использованием квантовых вычислений. Запускается процесс вычисления минимальной энергии с помощью метода VQE. В результате выполнения алгоритма получаем уточненное значение минимальной энергии системы. Уточненное значение минимальной энергии выводится на экран для оценки эффективности оптимизации. VQE позволяет эффективно итеративно находить минимальную энергию системы, что может быть использовано для оптимизации передачи энергии в системе с учетом энергетических затрат.

Результаты программы нахождения оптимальные маршруты передачи энергии между компонентами или устройствами внутри системы, используя алгоритм Дейкстры с учетом энергетических затрат приведены на рисунках 5.16 и 5.17.

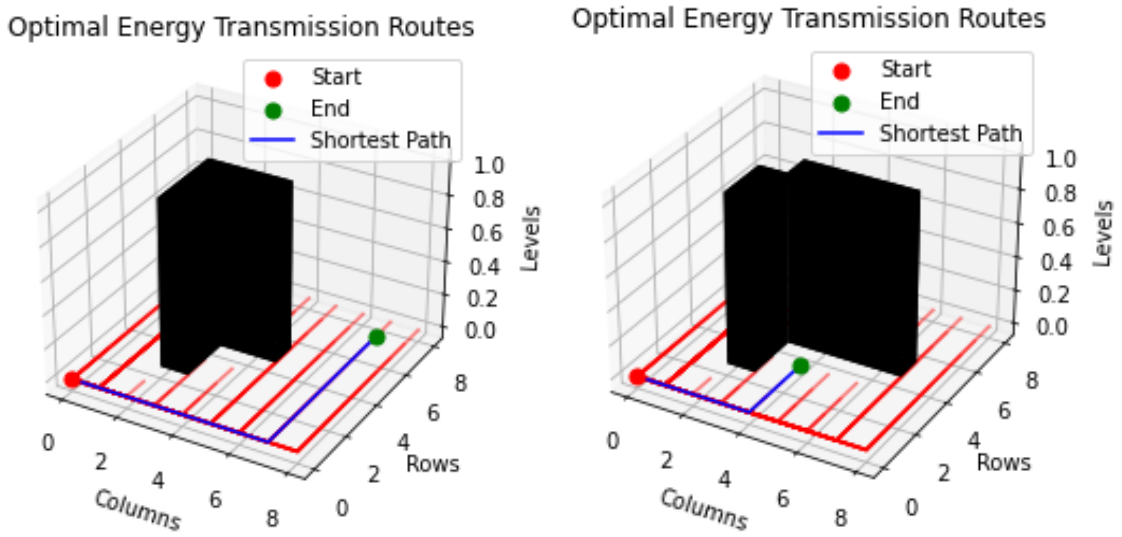


Рис.5.16. Оптимальные маршруты передачи энергии между компонентами или устройствами

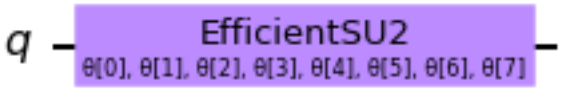


Рис.5.17. Квантовая схема оптимального маршрута передачи энергии между компонентами или устройствами

Это одна из вариационных форм квантовых схем, которая представляет собой слой, содержащий повороты вокруг оси Y (RY) и контролируемые операции Зейтенберга (CZ). Он принимает параметры (θ[0], θ[1], ..., θ[7]), которые представляют собой углы поворота вокруг оси Y для каждого кубита в квантовой схеме. Квантовая схема, состоит из восьми поворотов вокруг оси Y и контролируемых операций Зейтенберга между соответствующими кубитами. Эти параметры (θ[0], θ[1], ..., θ[7]) оптимизируются для минимизации энергии системы при использовании алгоритма оптимизации, такого как VQE.

QAOA (Quantum Approximate Optimization Algorithm) и VQE (Variational Quantum Eigensolver) - это два различных квантовых алгоритма, каждый из которых используется для разных целей и имеет свои уникальные особенности. Целью QAOA является решение задачи комбинаторной оптимизации путем поиска приближенного минимума (или максимума) целевой функции. Целью VQE является нахождение приближенного низшего собственного значения гамильтониана квантовой системы, что имеет применение в квантовых вычислениях и квантовой химии. QAOA применяется для задач комбинаторной оптимизации, таких как задача размещения, задача укладки, задача коммивояжера и другие. VQE применяется для решения задач квантовой химии, квантовых вычислений и других задач, связанных с квантовыми системами. QAOA имеет параметры, которые могут быть настроены для оптимизации производительности алгоритма для конкретной задачи оптимизации. VQEвключает в себя параметры вариационной формы, которые могут быть настроены для достижения оптимального приближения низшего собственного значения гамильтониана.

VQE позволяет выбирать различные вариационные формы для построения алгоритма, что делает его гибким инструментом для различных типов задач и систем. VQE часто обеспечивает более высокую точность результата, особенно при использовании большого

числа параметров вариационной формы и квантовых симуляторов высокой точности. Применение в квантовых вычислениях: VQE является ключевым компонентом в квантовых вычислениях, поскольку он используется для решения многих задач, таких как оптимизация портфелей, квантовая механика и другие. Эти преимущества делают VQE важным инструментом в квантовых вычислениях и квантовой химии, что дает ему преимущество перед QAOA в определенных сценариях и задачах. В целом, QAOA и VQE являются мощными инструментами в квантовых вычислениях, но каждый из них имеет свои уникальные особенности и применения в различных областях.

Разработка квантовых методов для решения задач оптимизации сложных энергетических систем представляет собой важное направление исследований, которое может иметь значительное воздействие на энергетическую отрасль и другие смежные области и открывает новые перспективы в решении задач, которые могут быть слишком сложны для классических методов. Вариационные квантовые алгоритмы, предоставляют эффективные инструменты для решения оптимизационных задач с большим числом переменных. В энергетике часто используются распределенные системы, и оптимизация таких систем представляет сложную задачу. Квантовые методы могут помочь учитывать множество переменных и ограничений, что может улучшить управление и эффективность распределенными системами. Разработка квантовых методов для оптимизации сложных энергетических систем обещает перевернуть традиционные методы оптимизации и создать более устойчивые и эффективные системы. Будущее этой области представляется ярким и полным возможностей.

Заключение

1. Актуальность использования методов мягких вычислений (Soft Computing), включающих такие компоненты, как нечеткие множества, нечеткая логика и нечеткий логический вывод, нейронные сети и эволюционные, в том числе генетические алгоритмы, обусловлена тем, что обработка данных на основе мягких вычислений способна справляться с неточностью, неопределенностью и частичной истинностью без потери производительности и эффективности для конечного использования. При этом технология Soft Computing предусматривает как самостоятельное, так и, как правило, совместное использование указанных выше ее компонент. Интеграция этих компонент дает возможность создавать широкий спектр гибридных интеллектуальных систем. Они позволяют решать многие важные проблемы реального мира, которые традиционными методами, в том числе методами искусственного интеллекта типа Hard Computing, решить практически невозможно.

2. Разработка квантовых методов для решения задач оптимизации представляет собой важное исследовательское направление, которое обладает потенциалом для революции в энергетической отрасли и смежных областях. Это направление сочетает в себе как фундаментальные аспекты квантовой физики, так и прикладные аспекты оптимизации и управления системами. Путем использования квантовых компьютеров и алгоритмов, а также с помощью симуляции квантовых систем, исследователи и инженеры могут более точно и эффективно решать задачи оптимизации, которые могли бы быть невыполнимы классическими методами. Квантовые методы предоставляют возможность учитывать множество переменных, ограничений и условий, что является критическим в условиях сложных энергетических систем. Несмотря на вызовы, такие как разработка квантовых аппаратов, управление шумами в квантовых системах и необходимость глубоких исследований, перспективы этой области блистательны.

Список использованной литературы

1. Алиев Р.А., Алиев Р.Р. Теория интеллектуальных систем. Учебное пособие. - Баку:Чашиоглы, 2001. – 720 с.
2. Devitt JS , Munro JW , Nemoto Kae (2011). High performance quantum computing, Special issue : Quantum information technology , No. 8, pp.49-55.
3. Emani P. S., Warrell J., Anticevic A., et al. Quantum computing at the frontiers of biological sciences. // Nat Methods. 2021. No. 18. P. 701–709.
4. Mangini S., et al. Quantum computing models for artificial neural networks. // EPL. 2021. P. 134.
5. Бочаров Н. А., Кирилюк М. А., Парамонов Н. Б. Квантовые вычисления и некоторые сложности их реализации // Приборы. 2021. № 7 (253). С. 18–25.
6. Chen H, Chaiang RHL, Storey VC (2012), Business intelligence and analytics : From big data to big impact, *MIS Quarterly, Special issue : Business intelligence research* , Vol. 36 No. 4.
7. Big Data Fundamentals: http://www.cse.wustl.edu/~jain/cse570-13
8. Quantum Artificial intelligence lab at NASA: www.nas.nasa.gov/quantum
9. Image source (1.4, 1.6, and 1.7): http://www.cqc2t.org/
10. Qubits and quantum measurements: http://www.inst.eecs.berkeley.edu/
11. Дивакар Майсор, Шрикант Кхупат, и Швета Джайн Архитектура и шаблоны больших данных. [Электронный ресурс] Режим доступа: https://www.ibm.com/developerworks/ru/library/bd-archpatterns1/index.html
12. Davy Cielen, Arno D. B. Meysman, and Mohamed Ali. Introducing Data Science. Big data, machine learning, and more, using Python tools https://www.manning.com/books/introducing-data-science
13. Hilbert, M. (2016). Big Data for Development: A Review of Promises and Challenges. Development Policy Review, 34(1), 135–174. http://doi.org/10.1111/dpr.12142
14. Environmental Protection Agency. Subpart A—General Provisions. https://www.gpo.gov/fdsys/pkg/CFR-2011-title40-vol1/pdf/CFR-2011-title40- vol1-part3-subpartA.pdf
15. Seth Gilbert and Nancy Lynch. Brewer’s Conjecture and the

Feasibility of Consistent, Available, Partition-Tolerant Web Services. https://doi.org/10.1145/564585.564601

16. Steven J. Plimpton and Karen D. Devine (2011), MapReduce in MPI for Large- scale Graph Algorithms, https://doi.org/10.1016/j.parco.2011.02.004

17. Инфосфера общественных наук России : монография / А. Б. Антопольский, Д. В. Ефременко ; под ред. В. А. Цветковой. – М. ; Берлин : Директ-Медиа, 2017. – 676 с.

18. Федеральный закон от 27 июля 2006 г. № 149-ФЗ “Об информации, информационных технологиях и о защите информации” с изменениями и дополнениями от: 27 июля 2010 г., 6 апреля, 21 июля 2011 г., 28 июля 2012 г., 5 апреля, 7 июня, 2 июля, 28 декабря 2013 г., 5 мая, 21 июля 2014 г. [Электронный ресурс] Режим доступа: http://base.garant.ru/12148555/

19. Hari Shreedharan, 2014, Using Flume: Flexible, Scalable, and Reliable Data Streaming ISBN-13: 978-1449368302

20. Neha Narkhede, Gwen Shapira, Todd Palino, 2017

21. Kafka: The Definitive Guide: Real-Time Data and Stream Processing at Scale ISBN-13: 978-1491936160

22. Д.И. Муромцев. Онтологический инжиниринг знаний в системе Protégé. – СПб: СПб ГУ ИТМО, 2007. – 62 с

23. Data is the new oil in the digital economy. https://www.wired.com/insights/2014/07/data-new-oil-digital-economy/

24. Data. Cambridge dictionary. https://dictionary.cambridge.org/dictionary/english/data

25. M.Rouse. Data Life Cycle. https://whatis.techtarget.com/definition/data-life-cycle

26. 26 Введение в нелинейное программирование. М.: Наука 1985.

27. Noll M.G. Running Hadoop on Ubuntu Linux (Multi-Node Cluster). http://www.michael-noll.com/tutorials/running-hadoop-on-ubuntu-linux-multi- node-cluster/

28. Гермейер Ю.Б. Введение в теорию исследования операций. М.: Наука, 1971.

29. Dresner Advisory Services, LLC. Big Data Analytics Market Study 2016. https://www.microstrategy.com/getmedia/cd052225-be60-49fd-ab1c-4984ebc3cde9/Dresner-Report-Big_Data_Analytic_Market_Study- WisdomofCrowdsSeries-2017

30. Приказ 11 февраля 2013 г. N 17 “При выводе из

эксплуатации машинных носителей информации, на которых осуществлялись хранение и обработка информации, осуществляется физическое уничтожение этих машинных носителей информации”:

31. Metadata Life Cycle. http://metadata.teldap.tw/design/lifecycle_eng.htm

32. Increasing Rate of Data Production Prompts Google to Rethink Data Center Storage. http://www.titanpower.com/blog/increasing-rate-of-data-production- prompts-google-to-rethink-data-center-storage/

33. Mellanox Scale-Out SN3000 Ethernet Switch Series. http://www.mellanox.com/page/products_dyn?product_family=280&mtag =sn300 0_label

34. Сысоев С.С. Введение в квантовые вычисления. Квантовые алгоритмы : учеб. пособие. – СПб. : Изд-во С.-Петерб. ун-та, 2019. – 144 с.

35. Квантовые вычисления : учеб. пособие. – Казань : Изд-во Казанского федерального университета, 2010. – 100 с.

36. Ожигов Ю.И. Квантовые вычисления : учеб.-метод. пособие. – М. : Изд-во Московского госуд. ун.-та, 2003. – 104 с.

37. Кайе Ф., Лафламм Р., Моска М. Введение в квантовые вычисления. – Москва– Ижевск : Регулярная и хаотическая динамика ; Институт компьютерных исследований, 2009. – 360 с.

38. Russian Quantum Center [Электронный ресурс]. – URL : https://www.youtube.com/channel/UCpOG8wlozPr6qXnO3kGoIxQ (дата обращения 10.10.2020).

39. Get started with IBM Quantum Experience [Электронный ресурс]. – URL: https://quantum-computing.ibm.com/docs (дата обращения 10.10/2020).

40. Зайченко Ю.П. Исследование операций: нечеткая оптимизация: Учеб. пособие.- Киев: Выща школа, 1991.- 191с.

41. Кабулов В.К. Алгоритмизация в социально-экономических системах. –Т.: Фан, 1989.

42. Круглов В.В., Дли М.И., Голунов Р.Ю. Нечеткая логика и искусственные нейронные сети. Физматлит, 2001. - 224 с

43. Калмыков С.А., Шокин Ю.И., Юлдашев З.Х. Методы интервального анализа. Новосибирск: Наука, 1986, 222с.

44. Кандель А., Байатт У.Дж. Нечеткие множества, нечеткая алгебра, нечеткая статистика. Труды американского общества инженеров-радиоэлектроников, т. 66, 1978, N12, с.37-61.

45. Камилов М.М., Акбаралиев Б.Б. Эвристический метод построения информативного признакового пространства в интеллектуальных системах анализа данных алгоритмами распознавания // Труды Восьмой Международный симпозиум «Интеллектуальные системы» (INTELS'2008), г.Нижний Новгород, Россия, -с. 113-116.

46. Кейн Л.А. Искусственный интеллект в обрабатывающих отраслях промышленности. Нефть, газ и нефтехимия за рубежом, N 9, 1986, с.117-122.

47. Кини Р.Л., Райфа Х. Принятие решений при многих критериях: предпочтения и замещения. - М: Радио и связь, 1981, 560с.

48. Кофман А. Введение в теорию нечетких множеств. М: Радио и связь, 1982, 432с.

49. Мелихов А.Н., Берштейн Л.С., Коровин С.Я. Ситуационные советующие системы с нечеткой логикой.- М.: Наука, 1990.- 272 с.

50. Мешалкин В.П. Экспертные системы в химической технологии. - М.: Химия, 1995. - 368 с.

51. Митюшкин Ю.И., Мокин Б.И., Ротштейн А.П. Soft-Computing: идентификация закономерностей нечеткими базами знаний.- Винница: УНІВЕРСУМ-Вінниця, 2002.- 145с.

52. Магомедов И.А. и др. Применение теории нечетких множеств к задачам управления нестационарными процессами. В сб.: Методы и системы принятия решений. - Рига: РПИ, 1984, с.60-65.

53. Малышев Н.Г., Берштейн Л.С., Боженюк А.В. Нечеткие модели для экспертных систем в САПР. - М.: Энергоатомиздат, 1991, 136с.

54. Мелихов А.Н. и др. Лингвистический терминальный комплекс. Модели выбора альтернатив в нечеткой среде. Рига: РПИ, 1984, с.140-142.

55. Мухамедиева Д.Т. Нечеткое многокритериальное динамическое моделирование в процессе принятия решений // Естественные и технические науки. –Москва. 2004. №2. С. 174-179.

56. Мухамедиева Д.Т. Задачи стохастического программирования в нечеткой среде // Актуальные проблемы современной науки. –Москва, 2004, №3, с. 175-180.

57. Мухамедиева Д.Т. Статическая модель принятия решений в условиях неопределенности // Вестник ТашГТУ». –Ташкент, 2007. Вып.1. Стр. 57-61.

58. Мухамедиева Д.Т. Статистическое моделирование в сельском хозяйстве с применением теории нечетких множеств. - Ташкент: Институт кибернетики НТЦ «Современные информационные технологии». 2004. –200 с.

59. Мухамедиева Д.Т. Моделирование слабо формализуемых процессов на основе обработки нечеткой информации.-Ташкент: Институт информатики АН РУз, 2007. -231 с.

60. Muhamediyeva D.T. Noravshan axborot holatida sust shakllangan jarayonlarni modellashtirish. -Toshkent. Matematika va axborot texnologiyalar instituti. 2010. 400 bet.

61. Muhamediyeva D.T. Noravshan axborotni qayta ishlash asosida sust shakllangan jarayonlarni tizimli modellashtirish muammolari. - Toshkent. Matematika va axborot texnologiyalar instituti. 2010. 531 bet.

62. Насибов Э.Н. Об идентификации состояний объекта при нечеткой информации // Междунар. конф. по мягким вычислениям и измерениям: SCM’2001. –Санкт-Петербург,2001. –1. –С.151-154.

63. Насибов Э.Н. Методы обработки нечеткой информации в задачах принятия решений. –Баку: Элм, 2000. –260с.

64. Недосекин А.О. Нечетко-множественный анализ риска фондовых инвестиций. СПб: Изд-во Сезам, 2002. - 181 с.

65. Нечеткие множества в моделях управления искусственного интеллекта /Под ред. Д.А. Поспелова. –М.: Радио и связь, 1987. –330с.

66. Несенюк А.П. Неопределенные величины в задачах управления с неполной информацией. - Автоматика, N 2, 1979, с.55-64.

67. Нечеткие множества и теория возможностей. Последние достижения. - М: Радио и связь, 1986, 408с.

68. Норвич А.М., Туркcен И.Б. Построение функций принадлежности. В сб.: Нечеткие множества и теория возможностей. - М: Радио и связь, 1986, с.64-71.

69. Норвич А.М., Туркcен И.Б. Фундаментальное измерение нечеткости. В сб.: Нечеткие множества и теория возможностей. - М: Радио и связь, 1986, с.54-64.

70. Орлов А.И. Задачи оптимизации и нечеткие переменные.-М.: Знание, 1980.- 64 с.

71. Орлов А.И. Связь между средними величинами и допустимыми преобразованиями шкалы.- Математические заметки, т. 30, вып. 4, 1981, с. 561-568.

72. Осуга С. Обработка знаний. - М.: Мир, 1989. - 293 с.

73. Орловский С.А. Проблемы принятия решений при нечеткой исходной информации. М: Наука, 1981, 203с.

74. Подиновский В.В., Ногин В.Д. Парето-оптимальные решения многокритериальных задач. М.: Наука, 1982.

75. Поспелов Д.А. Логико-лингвистические модели в системах управления.- М.:Энергоиздат, 1981.- 232 с.

76. Поспелов Д.А. Ситуационное управление: теория и практика.- М. Наука, 1986.- 288 с.

77. Поспелов Г.С. и др. Процедуры и алгоритмы формирования комплексных программ. М: Наука, 1985, 424с.

78. Потюлкин А.Ю. Решение задач идентификации нечетких систем // Изв. АН России. Теория и системы управления. –1996. -№4. –С.40-46.

79. Прикладные нечеткие системы/Асаи К., Ватада Д., Иваи С. и др./Под ред. Т. Тэрано, К. Асаи, М. Сугено.- М.: Мир, 1993. - 368 с.

80. Пшеничный Б.Н.. Данилин Ю.М. Численные методы в экстеримальных задачах. М.: Наука, 1975.

81. Райская Н.Н., Френкель А.А. Применение гребневой регрессии в статистическом моделировании // Экономика и мат. методы, 1985. Т. XXI. Вып. 4.

82. Ротштейн А.П. Интеллектуальные технологии идентификации: нечеткая логика, генетические алгоритмы, нейронные сети. — Винница: УНИВЕРСУМ—Винница, 1999. — 320 с.

83. Ротштейн А.П. Медицинская диагностика на нечеткой логике. — Винница: Континент—ПРИМ, 1996. — 132 с.

84. Ротштейн А.П., Штовба С.Д. Нечеткий многокритериальный анализ вариантов с применением парных сравнений // Известия РАН. Теория и системы управления.- 2001.- №3.- С.150-154.

85. Рахматуллаев М.А. Теория и прикладные методы построения метасистемы генерации решений в детерминированной и нечеткой технологической среде: Автореф. дис….докт. техн. наук., -Ташкент, 1994. –38 с.

86. Рыбкин В.А., Язенин А.В. О сильной устойчивости в задачах возможностной оптимизации // Изв. РАН ТИСУ. 2000. -№2.

87. Рутковская Д., Пилинский М., Рутковский Л. Нейронные сети, генетические алгоритмы и нечеткие системы: Пер.с польск. И.Д. Рудинского. – М.: Горячая линия – Телеком, 2004. – 452 с.

88. Саттаров Д. Сорт, почвы, удобрение и урожай. –Т.: Мехнат. 1988.

89. Семухин М.В. Разрешимость нечетких и интервальных уравнений. Вестник Тюменского государственного университета, вып.2. - Тюмень, ТюмГУ, 1998, с.23-26.

90. Семухин М.В. Теория нечетких множеств. Учебно-методическое пособие. - Тюмень: ТюмГУ, 1999, 50 с.

91. Степанов В.В. Численное решение некорректно поставленных задач на множествах кусочно-монотонных и выпуклых функций // Вести МГУ. Сер.15. -1985, №3.-С.21-26.

92. Тихонов А.Н., Арсенин В.Я. Методы решения некорректных задач. -М.: Наука, 1986.

93. Уткин Л.В., Шубинский И.Б. Нетрадиционные методы оценки надежности информационных систем. – СПб.: Любавич, 2000. – 173 с.

94. Фиакко А, Мак-Кормик Г. Нелинейное программирование. Методы последовательной безусловной оптимизации. М.: Мир, 1972.

95. Химмельблау Д. Анализ процессов статистическими методами. - М: Мир, 1973, 468с.

96. Хинтон Д.Е. Как обучаются нейронные сети // В мире науки. – 1992. - № 11-12. – С.103-110.

97. Чуклеев С.Н. К вопросу о разрешимости нечетких уравнений. В сб.: Модели выбора альтернатив в нечеткой среде. Рига: РПИ, 1984, с.95-96.

98. Цыпкин Я.З. Основы информационной теории идентификации. -М.: Наука,1984.-320с.

99. Шокин И.Ю. Интервальный анализ. Новосибирск: Наука, 1981, 112 с.

100. Ягер Р.Р. Множества уровня для оценки принадлежности нечетких подмножеств. В сб.: Нечеткие множества и теория возможностей. М: Радио и связь, 1986, с.71-78.

101. Язенин А.В. Некоторые методы теории нечетких подмножеств в многокритериальных задачах с приложением. Автореф. дис. на соис. учен. степ. канд. техн. наук.- Рига: Риж. политех. ин-т. 1982. -22 с.

102. Bellman R., Kalaba K., Zadeh L.A. Abstraction and pattern classification. J.Math. Anal. and Appl., v.13, No1, Jan, 1966.

103. Bellman R.E., Gierts M. On the analitical formalism of theory of fuzzy sets."Inform. Sci.", 1973, v.5, N2, p.149-156.

104. Bonissone P.P., Tong R.M. Editorial: reasoning with uncertainty in expert systems."Int. J. Man-Mach. Stad.", 1985, N3, p.241-250.

105. Bekmuratov T.F., Mukhamedieva D.T. Decision-making problem in poorly formalized processes. // Proceedings of WCIS-2008, b –Quadrat Verlag. 2008. P. 214-218.

106. Bekmuratov T.F., Muhamedieva D.T., Bobomuradov O.J. Model prediction of yield with fuzzy initial conditions / Proceedings of Ninth International Conference on Application of Fuzzy Systems and Soft Computing ICAFS, Prague, Czech Republic, 2010,-p.321-328.

107. Chang S.S.L. Application of fuzzy set theory to economics. "Kybernetes", 1977, v.6, p.203-208.

108. Daley S., Gill K.F. The fuzzy logic controller: an alternative design scheme? "Comput. Ind.", 1985, N1, p.3-14.

109. Dubois D., Prade H. Operations on fuzzy numbers. Int. J.System sci., 1978, v.5, N2, p.613-626.

110. Dubois D., Prade H. Fuzzy sets and systems: Theory and applisations. - New York: Acad. Press, 1980, 394p.

111. Dubois D., Prade H. Fuzzy real algebra: Some rezults. - "Fuzzy Sets and Systems". 1979, v.2, N4, p.327-348.

112. Dubois D., Prade H. Systems of linear fuzzy constraints. - "Fuzzy Sets and Systems". 1980, v.3, N1, p.37-48.

113. Dubois D., Prade H. Inverse operations for fuzzy numbers.-In: Proc. of IFAC Symp. “Fuzzy Information, Knowledge Representation, and Decision Analysis”. Marceille: IFAC, 1983, p.34-48.

114. Freeling A.N.S. Fuzzy sets and decision analysis. "IEEE Tran. Syst.Man. and Cybern.", v. 10, N7, p.341-354.

115. Flondor P. An example a fuzzy system. "Kybernetes". 1977, p.229-230.

116. Funy J.W., Fu K.S. An axiomatic approach to rational decision making in a fuzzy environment. "Fuzzy Sets and Their Application to Cognitive and Decision Processes", New York, 1975, p.227-257.

117. Gaines B.R. Stochastic and fuzzy logical. "Electron. Lett.", v. 11, 1975, p188-189.

118.Garlott J. Interval Mathematics. A Bibliography. Freiburger, Interval-Berichte, West Germany, N6, 1985, 250p.

119. Gen M., Cheng R. Genetic algoritms and engineering design. – John Wiley&Sons, 1997. – 352 p.

120. Goguen Y.A. The logic of inexact concepts. "Synthese", v. 19, p.329-373.

121. Golden B.L. Nonlinear programming on a microcomputer. "Comput. and Oper. Res.", 1986, N2-3, p.149-166.

122. Gorzalczany M.B. Interval-Valued Decisional Rule in Signal Transmission Problems. "Arhiwum automatyki i telemechaniki", t.XXX, N2, 1985, p.159-168.

123. Govind R. Synthesis of fuzzy controllers for process plants. "Proc. Int. Conf. Cybern. and Soc., Tokio-Kyoto", New York, 1978, v.2-3, p.1228-1232.

124. Hopfield J., Tank D., Neural computation of decision in optimization problems, Biol. Cybernet, 1985, vol.52, pp. 141-152.

125. Hung D.L. Wang J. Digital hardware realization of a recurrent neural network for solving the assignment problem // Neurocomputing, 51, 2003, pp. 447-461.

126. Johnson R.W., Shore J.E. Solving Fuzzy Sets Problems Using Probability Theory, Technical Memorandum NRL 7503-211, Naval Research Laboratory, Washington, D.C., 1979.

127. Kickert W.Y.M. and oth. Application of Fuzzy Controller in a Warm Water Plent. "Automatica", v. 12, N4, 1976, p.301-308.

128. Kelsey J. Immune Inspired Somatic Contiguous Hypermutation for Function Optimisation / J. Kelsey, J. Timmis // Proc. of Genetic and Evolutionary Computation Conference. Springer Lecture Notes in Computer Science 2723. – 2003. –P. 207-218.

129. Mamdani E.H., Efstathion H.J. Higher-order logics for handling uncertainty in expert systems. "Int. J. Man-Mach. Stud.", 1985, N3, p.243-259.

130. Markov S.M. Extended interval arithmetics.- Докл. Болг. АН, 1977, т. 30, № 9, с.1239-1242.

131. Muhamedieva D.T. To one algoritm of finding of analitical deciding a nonlinear programming in fuzzy ambience // Proceedings of WCIS-2004, b –Quadrat Verlag. 2004. P. 94-98.

132. Moor R.E. A servey of interval methods for differential equations. "Proc. 23_rd_ IEEE Conf. Decis. and Contr., Las Vegas, Nev., 1984, v.3", New York, 1984, p.1529-1535.

133. Oden G.S. Integration of fuzzy logical information .- "J. Exp. Psychol.", 1977, v.3, N4, p.505-575.

134. Ogawa Hideo. Labeled point pattern matching by fuzzy relaxation. "Pattern.Recogn.", 1984, v.17, N5, p.569-573.

135. Ostermark R. Sensitivity analysis of linear fuzzy programs: an approach to parametric interdepedence. "Kybernetes", 1987, N2, p.113-120.

136. O'Keefe R. Simulation and experts systems - a taxsonomy and some examples. "Simulation", 1986, 46, N1, p.10-16.

137. Pal S.K., Majumaer D.D. Effect of fuzzyfication on the plosive cognition system. "Int. J. Systems Sci.", 1978, v.9, N8, p.873-886.

138. Pavlak Z. Roufh sets and fuzzy sets. "Pr. IPI PAN", 1984, N540, 10p.

139. Prade H. A computional approach to approximate and plausible reasoning with applications to expert systems. "IEEE Trans. Pattern Anal. and Mach. Intel.", 1985, N3, p.260-283.

140. Saaty T.L. Multicriteria decision making: the analytical hierarchy process. N.Y.: McGraw Hill, 1990. –502 p.

141. Saaty T.L. The analytic network process. –Pittsburgh: RWS Publ., 1996. –370 p.

142. Schwandt H. Newton-like interval methods for large nonlinear systems of equations on vector computers. "Comput. Phys. Commun.", 1985, N1-3, p.223-232.

143. Sugeno M. Fuzzy measure and fuzzy integral. -Trans. SICE, 1972, v.8, № 2, -p. 95-102.

144. Shtovba S., Rotshtein A., Pankevich O. Fuzzy Rule Based System for Diagnosis of Stone Construction Cracks of Buldings. In "Advances in Computational Intelligence and Learning, Methods and Applications" (Editors: Zimmermann H-J., Tselentis G., van Someren M., Dounias G.): Kluwer Academic Publishers, 2001, P.401-412.

145.Tanaka H., Asai K. Fuzzy solution in fuzzy linear pogremming problems. "IEEE Trans. Syst. Maan and Cybern.", 1984, N2, p.325-328.

146. Tozan H., Vayvay O. Analyzing Demand Variability Through SC Using Fuzzy Regression and Grey GM(1,1) Forecasting Models, Information Sciences 2007, World Scientific, 2007, pp. 1088-1094.

147. H.Tozan, M.Yagimli A. Fuzzy Prediction Based Trajectory Estimation. // Wseas transactions on systems. Issue 8, vol.-9, august 2010. pp.885-894.

148. Urban B., Hansel V. A fuzzy concept in the theory of strategic decision where several objectives exist. "Fuzzy inf., Proc. IFAC Symp. Marseille, 19-21 July, 1983." Oxford e.a., 1984, p.313-320.

149. Willaeys D. Some of the properties of fuzzy discretisation. "Fuzzy Inf., IFAC Symp. Marseille, 19-21 July, 1983." Oxford, 1984, p.61-69.

150. Yamazaki T., Sugeno M. Самоорганизующийся нечеткий регулятор. "Кэйсоку дзидо сэйге гаккай ромбунсю,Trans. Soc. Instrum. and Contr. Eng.", 1984, N8, p.720-726.

151. Goldberg D. E. Genetic Algorithms in Search, Optimization and Machine Learning. Reading: Addison Wesley, New York, 1989 - 412p.

152. Батищев Д.И. Генетические алгоритмы решения экстремальных задач // Учеб. пособие. Воронеж, 1995. - 69 с.

153. Курейчик В.М. Генетические алгоритмы. Состояние, проблемы, перспективы // Известия Академии наук. 1999. - №1. - С. 144-160.

154. Holland J.H. Adaptation in Natural and Artificial Systems // Univ.of Michigan Press, Ann Arbor. 1975.

155. Ramsey C.L., Grefenstette J.J. Case-based Initialization of Genetic Algorithms // 5th Int. Conf. on Genetic Algorithm. 1993. - P.84-91.

156. Szpiro G.G. Forecasting Chaotic Time Series with Genetic Algorithms // Physical Review E. 1997. - V.55, N3. - P. 2557-2568.

157. Alvarez A. Forecasting the SST Space-Time Variability of the Alboran Sea with Genetic Algorithms // Geophysical Research Letters. 2000.

158. Загоруйко Н.Г. Самообучающийся генетический алгоритм для прогнозирования. // Искусственный интеллект и экспертные системы. Новосибирск, 1997. - Вып. 160, "Вычислительные системы". - С. 80-95.

159. Louis S.J., Xu Z. Genetic Algorithms for Open Shop Scheduling and ReScheduling // ISC A 11th Int. Conf. on Computers and their Applications. -1996. P.99-102.

160. Davidor Y. A. Genetic Algorithm Applied to Robot Trajectory Generation // Parallel Problem Solving from Nature 4. 1996.89

161. Zalzala A.M.S., Fleming P.S. Genetic Algorithms: Principles and Applications in Engineering Systems // Neural Network. 1996. - Vol. 6, N5. -P.803-820.

162. Ono O., Kobayashi B., Kato H. Optimal Dynamic Motion Planning of Autonomous Vehicles by a Structured Genetic Algorithm // Proc. of the 13th World Congress of IFAC, vol. Q. San-Francisco, USA,1996. -P.435-440.

163. Forrest S. Mayer-Kress G. Genetic Algorithms, Nonlinear Dynamical Systems and Models of International Security // Parallel Problem Solving from Nature 4. 1996.

164. Cobb H.G. An Investigation into the Use of Hypermutation as an Adaptive Operator in the Genetic Algorithm Having Continuous Time-Dependent Nonsationary Environments. Naval Research Laboratory Memorandum Report 6760. - 1990.

165. Vavak F., Fogarty T.C., Jukes K. A Genetic Algorithm with Variable Range of Local Search for Tracking Changing Environments // Parallel Problem Solving from Nature 4. 1996.

166. Ярушкина Н.Г. Основы теории нечетких и гибридных систем. Учебное пособие. – М.: Финансы и статистика, 2004.

167. Zadeh L.A. Toward a theory of fuzzy information granulation and its centrality in human reasoning and fuzzy logic. // Fuzzy sets and systems. – Vol.90. – 1997. - № 2.

168. Fanabashi M., Maeda A., Morooka Y., Mori K. Fuzzy and Neural Hybrid Systems: Synergetic AI // IEEE Expert. 1995 August. – p. 32 – 40.

169. Кушербаева В.Т. Некоторые тестовые функции для глобальной оптимизации. – СПб.: СПбГУ. 2007.

170. Herrera F., Lozano M. Adaptation of genetic algorithm parameters based on fuzzy logic controllers. In: F. Herrera, J. L. Verdegay (eds.) Genetic Algorithms and Soft Computing, Physica-Verlag, Heidelberg, 1996. - pp. 95-124.

171 Емельянов В.В., Курейчик В.В., Курейчик В.М. Теория и практика эволюционного моделирования. – М.: Физматлит, 2003.

172. Голубин А.В., Тарасов В.Б. Нечеткие генетические алгоритмы. // Международные научно-технические конференции AIS’05 и CAD-2005. Труды конференций. – М.: Физматлит, 2005. – с. 39 – 45.

173. Kalyanmoy D., Dhiraj J., Ashish A. Real-Coded Evolutionary Algorithms with Parent-Centric Recombination. Indian Institute of Technology, Kanpur. KanGAL Report Nº 2001003.

174. Lozano M., Herrera F., Krasnogor N., Molina D. Real-Coded Memetic Algorithms with Crossover Hill-Climbing. Evolutionary Computation 12(3), 2004. – p. 273 – 302.

175. Herrera F., Lozano M., Sanchez A.M. A Taxonomy for the Crossover Operator for Real-Coded Genetic Algorithms: An Experimental

Study. // International Journal of Intelligent Systems, vol. 18, 2003. – p. 309 – 338.

176. Herrera F., Lozano M. Fuzzy Adaptive Genetic Algorithms: design, taxonomy, and future directions. // Soft Computing 7(2003), Springer-Verlag, 2003. – p. 545 – 562.

177. Galantucci L.M., Percoco G., Spina R. Assembly and Disassembly Planning by using Fuzzy Logic & Genetic Algorithms. // International Journal of Advanced Robotic Systems, Vol. 1, № 2, 2004. – p. 67 – 74.

178. Гладков Л.А. Алгоритм выделения ядер в нечетких графах на основе моделирования эволюции. IX национальная конференция по искусственному интеллекту с международным участием КИИ'2004. Труды конференции. – М.: Физматлит, 2004. с. 346 - 355.

179. Finkelstein A.V., Gutin A.M., Badretdinov A.Y //Proteins. - 1995. -V.23. -P.P.151-162.

180. Tarakanov A.O. Formal peptide as a basic of agent of immune networks: from natural prototype to mathematical theory and applications //Proceeding of the I Int. workshop of central and Eastern Europe on Multi - Agent Systems, 1999. -P.P.186-188.

181. Dasgupta D., Artificial Immune Systems and Their Applications, Springer-Verlag, 1998.

182. Gaber J., Bakhouya M., An Immune Inspired-based Optimization Algorithm: Application to the Traveling Salesman Problem, Université de Technologie de Belfort-Montbéliard, 2007.

183. Искусственные иммунные системы и их применение / Под ред. Д. Дасгупты; пер. с англ. под ред. А.А. Романюхи. – М.: ФИЗМАТЛИТ, 2006. – 344 с.

184. An Overview of Artificial Immune Systems / J.I. Timmis, T. Knight, L.N. De Castro, E.H. Art // Computation in Cells and Tissues: Perspectives and Tools for Thought, Natural Computation Series, Springer, 2004. – P. 51-86.

185. De Castro L.N. Learning and optimization using the clonal selection principle / L.N. De Castro, F.J. Von Zuben // IEEE Transactions on Evolutionary Computation, Special Issue on Artificial Immune Systems. – 2002. – P. 239-251.

186. Kelsey J. Immune Inspired Somatic Contiguous Hypermutation for Function Optimisation / J. Kelsey, J. Timmis // Proc. of Genetic and Evolutionary Computation Conference. Springer Lecture Notes in Computer Science 2723. – 2003. –P. 207-218.

187. Villalobos-Arias M. Convergence Analysis of a Multiobjective Artificial Immune System Algorithm / M. Villalobos-Arias, C.A. Coello Coello, O. Hernández-Lerma // In Proc. ofICARIS 2004, Springer Lecture Notes in Computer Science 3239. – P. 226-235.

188. De Castro L.N. AiNet: an artificial immune network for data analysis in Data Mining / L.N. De Castro, F.J. Von Zuben // A Heuristic Approach, Chapter XII, H.A. Abbass, R.A. Sarker, and C.S.Newton, Eds. USA: Idea Group Publishing. – 2001 – P. 231-259.

189. Штовба С.Д. Муравьиные алгоритмы // Exponenta Pro. Математика в приложениях. – 2003. – №4. – С.70-75.

190. Clonal Selection Algorithms / V. Cutello, G. Narzisi,G. Nicosia, M. Pavone // A Comparative Case Study using Effective Mutation Potentials, ICARIS 2005, LNCS, Springer. –2005. – Vol. 3627. – P. 13-28.

191. Kaleva, O. Fuzzy differential equations / O. Kaleva // Fuzzy Sets and Systems. — 1987. — Vol. 24, № 3. — P. 301 — 317.

192. Комлева, Т.А. Усреднение нечетких дифференциальных уравнений / Т.А. Комлева, А.В. Плотников, Л.И. Плотникова // Тр. Одес. политех. ун-та. — Одесса, 2007. — Вып. 1(27). — С. 185 — 190.

193. Kaleva, O. The Peano theorem for fuzzy differential equations revisited / O. Kaleva // Fuzzy Sets and Systems. — 1998. — № 98. — P. 147 — 148.

194. Kaleva, O. O notes on fuzzy differential equations / O. Kaleva // Nonlinear Analysis. — 2006. — № 64. — P. 895 — 900.

195. Park, J.Y. Existence and uniqueness theorem for a solution of fuzzy differential equations / J.Y. Park, H.K. Han // Internat. J. Math. and Math. Sci. — 1999. — Vol. 22, № 2. — P. 271 — 279.

196. Park, J.Y. Fuzzy differential equations / J.Y. Park, H.K. Han // Fuzzy Sets and Systems. — 2000. — № 110. — P. 69 — 77.

197. Puri, M.L. Differential of fuzzy functions / M.L. Puri, D.A. Ralescu // J. Math. Anal. Appl. — 1983. — № 91. — P. 552 — 558.

198. Hukuhara, M. Integration des applications mesurables dont la valeur est un compact convexe / M. Hukuhara // Func. Ekvacioj. — 1967. — № 10. — P. 205 — 223.

199. Aubin, J.P. Fuzzy differential inclusions / J.P. Aubin // Problems of Control and Information Theory. — 1990. — Vol.19, № 1. — P. 55 — 67.

200. Baidosov, V.A. Differential inclusions with fuzzy right-hand side / V.A. Baidosov // Soviet Mathematics. — 1990. — Vol. 40, № 3. — P. 567 — 569.

201.Дойч Д. 2015. Структура реальности. Наука о параллельных вселенных. М.: Альпина нон-фикшн.

202.Кайе Ф.: Введение в квантовые вычисления. - Ижевск: Институт компьютерных исследований, 2009

203.Дойч Д. 2015. Начало бесконечности. Объяснения, которые меняют мир. М.: Альпина нон-фикшн.

204.Гринштейн Дж.: Квантовый вызов. - Долгопрудный: Интеллект, 2008

205.Коноплев Ю. М., Сысоев С. С. Симулятор квантового компьютера. http://qc-sim.appspot.com (дата обращения: 11.03.2019).

206.Валиев К.А.: Квантовые компьютеры: надежды и реальность. - М. ; Ижевск: Регулярная и хаотическая динамика, 2002

207. IBM Quantum Experience [Электронный ресурс] – URL : https://quantumcomputing.ibm.com/composer/new-experiment (дата обращения 10.10.2020).

208. Сысоев С.С. Введение в квантовые вычисления. Квантовые алгоритмы : учеб. пособие. – СПб. : Изд-во С.-Петерб. ун-та, 2019. – 144 с.

209.Deutsch D. 1985. Quantum theory, the Church-Turing principle and the universal quantum computer. Proceedings of the Royal Society A mathematical physical and engineering Sciences 400: 97–117.

210. Квантовые вычисления : учеб. пособие. – Казань : Изд-во Казанского федерального университета, 2010. – 100 с.

211.Garey M. R., Johnson D. S. 1979. Computers and Intractability: A Guide to the Theory of NP-Completeness. W. H. Freeman and Co.

212. Ожигов Ю.И. Квантовые вычисления : учеб.-метод. пособие. – М. : Изд-во Московского госуд. ун.-та, 2003. – 104 с.

213.Кайзер С., Гранад К. К15 Изучаем квантовые вычисления на Python и Q# / пер. с анг. А. В. Логунова. – М.: ДМК Пресс, 2021. – 430 с.

214. Get started with IBM Quantum Experience [Электронный ресурс]. – URL: https://quantum-computing.ibm.com/docs (дата обращения 10.10/2020).

215. Кайе Ф., Лафламм Р., Моска М. Введение в квантовые вычисления. – Москва– Ижевск : Регулярная и хаотическая динамика ; Институт компьютерных исследований, 2009. – 360 с.

216. Russian Quantum Center [Электронный ресурс]. – URL : https://www.youtube.com/channel/UCpOG8wlozPr6qXnO3kGoIxQ (дата обращения 10.10.2020).

217. Preskill J. Lecture notes for “Physics 219/Computer Science 219. Quantum Computation” (Formerly Physics 229). http://www.theory.caltech.edu/people/preskill/ph229/index.html (accessed 11.03.2019).

218. Кайзер С., Гранад К. К15 Изучаем квантовые вычисления на Python и Q# / пер. с анг. А. В. Логунова. – М.: ДМК Пресс, 2021. – 430 с.

219. J. Murray. Mathematical Biology: I. An Introduction. Third edition. Springer, Berlin, 2007.

220 .A. Okubo and S. Levin. Diffusion and Ecological Problems: Modern Perspec-tives. Second edition. Springer, New York, 2002.

221. D. Headley, T. Muller, A. Martin, E. Solano, M. Sanz, and F. K. ¨ Wilhelm, Approximating the quantum approximate optimisation algorithm, arXiv preprint arXiv:2002.12215 (2020).

222. N. Killoran, T. R. Bromley, J. M. Arrazola, M. Schuld, N. Quesada, and S. Lloyd, Continuous-variable quantum neural networks, Physical Review Research 1, 033063 (2019).

223. D. Wecker, M. B. Hastings, and M. Troyer, Progress towards practical quantum variational algorithms, Physical Review A 92, 042303 (2015).

224. J. Biamonte, P. Wittek, N. Pancotti, P. Rebentrost, N. Wiebe, and S. Lloyd, Quantum machine learning, Nature 549, 195 (2017).

225. L. Zhou, S.-T. Wang, S. Choi, H. Pichler, and M. D. Lukin, Quantum approximate optimization algorithm: Performance, mechanism, and implementation on near-term devices, Physical Review X 10, 021067 (2020).

226. K. Bharti, A. Cervera-Lierta, T. H. Kyaw, T. Haug, S. AlperinLea, A. Anand, M. Degroote, H. Heimonen, J. S. Kottmann, T. Menke, W.-K. Mok, S. Sim, L.-C. Kwek, and A. AspuruGuzik, Noisy intermediate-scale quantum algorithms, Reviews of Modern Physics 94, 015004 (2022).

227. S.-H. Noh, Analysis of gradient vanishing of rnns and performance comparison, Information 12 (2021)

228. E. Farhi and A. W. Harrow, Quantum supremacy through the quantum approximate optimization algorithm, arXiv preprint arXiv:1602.07674 (2016).

229. D. Guery-Odelin, A. Ruschhaupt, A. Kiely, E. Torrontegui, ´ S. Mart´ınez-Garaot, and J. G. Muga, Shortcuts to adiabaticity: Concepts, methods, and applications, Reviews of Modern Physics 91, 045001 (2019).

230. E. Torrontegui, S. Ibanez, S. Mart ´ ´ınez-Garaot, M. Modugno, A. del Campo, D. Guery-Odelin, A. Ruschhaupt, X. Chen, ´ and J. G. Muga, Shortcuts to adiabaticity, Advances in atomic, molecular, and optical physics 62, 117 (2013).

231 X. Chen, A. Ruschhaupt, S. Schmidt, A. del Campo, D. Guery- ´ Odelin, and J. G. Muga, Fast optimal frictionless atom cooling in harmonic traps: Shortcut to adiabaticity, Physical Review Letters 104, 063002 (2010).

232 X. Chen, E. Torrontegui, and J. G. Muga, Lewis-riesenfeld invariants and transitionless quantum driving, Physical Review A 83, 062116 (2011).

233. L. Zhu, H. L. Tang, G. S. Barron, F. Calderon-Vargas, N. J. Mayhall, E. Barnes, and S. E. Economou, An adaptive quantum approximate optimization algorithm for solving combinatorial problems on a quantum computer, arXiv preprint arXiv:2005.10258 (2020).

234. S. Masuda and K. Nakamura, Fast-forward of adiabatic dynamics in quantum mechanics, Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences 466, 1135 (2010).

235. M. Demirplak and S. A. Rice, Adiabatic population transfer with control fields, The Journal of Physical Chemistry A 107, 9937 (2003).

236. J. Weidenfeller, L. C. Valor, J. Gacon, C. Tornow, L. Bello, S. Woerner, and D. J. Egger, Scaling of the quantum approximate optimization algorithm on superconducting qubit based hardware, arXiv preprint arXiv:2202.03459 (2022).

237. M. V. Berry, Transitionless quantum driving, Journal of Physics A: Mathematical and Theoretical 42, 365303 (2009).

238. A. del Campo, Shortcuts to adiabaticity by counterdiabatic driving, Physical Review Letters 111, 100502 (2013).

239. K. Takahashi, Hamiltonian engineering for adiabatic quantum computation: Lessons from shortcuts to adiabaticity, Journal of the Physical Society of Japan 88, 061002 (2019).

240 S. Hadfield, Z. Wang, B. O'Gorman, E. G. Rieffel, D. Venturelli, and R. Biswas, From the quantum approximate optimization algorithm to a quantum alternating operator ansatz, Algorithms 12 (2019).