Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет
имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ

КАФЕДРА СИСТЕМЫ ОБРАБОТКИ ИНФОРМАЦИИ И УПРАВЛЕНИЯ

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К НАУЧНО-ИССЛЕДОВАТЕЛЬСКОЙ РАБОТЕ

НА ТЕМУ:

Применение машинного обучения для коррекции ошибок в квантовых вычислениях

Студент ИУ5-35М _____ _____ Р.А. Кузьмин
(Группа) (Подпись, дата) (И.О.Фамилия)

Руководитель _____ Ю.Е. Гапанюк
(Подпись, дата) (И.О.Фамилия)

2024 г.

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

УТВЕРЖДАЮ
Заведующий кафедрой ___ИУ5____
(Индекс)
_____ ___В.И. Терехов__
(И.О.Фамилия)
«_04_» ____сентября_____ 2024 г.

З А Д А Н И Е
на выполнение научно-исследовательской работы

по теме _____Применение машинного обучения для коррекции ошибок в квантовых_____
_____вычислениях_____

Студент группы _ИУ5-35М__

_____Кузьмин Роман Александрович_____
(Фамилия, имя, отчество)

Направленность НИР (учебная, исследовательская, практическая, производственная, др.)
_____ИССЛЕДОВАТЕЛЬСКАЯ_____
Источник тематики (кафедра, предприятие, НИР) _____КАФЕДРА_____

График выполнения НИР: 25% к ____ нед., 50% к ____ нед., 75% к ___ нед., 100% к ____ нед.

Техническое задание ___Рассмотреть теоретические и практические аспекты применения машинного обучения для борьбы с ошибками в квантовых вычислениях, сравнить существующие подходы_____
_____

Оформление научно-исследовательской работы:

Расчетно-пояснительная записка на _20__ листах формата А4.
Перечень графического (иллюстративного) материала (чертежи, плакаты, слайды и т.п.)
_____
_____
_____
_____

Дата выдачи задания «_04_» ____сентября_____ 2024 г.

Руководитель НИР _____ _____Ю.Е. Гапанюк__
(Подпись, дата) (И.О.Фамилия)

Студент _____ _____Р.А. Кузьмин_____
(Подпись, дата) (И.О.Фамилия)

Примечание: Задание оформляется в двух экземплярах: один выдается студенту, второй хранится на кафедре.

СОДЕРЖАНИЕ

1.

1. ВВЕДЕНИЕ

Калибровка квантового компьютера представляет собой процесс настройки его компонентов и параметров с целью обеспечения точного и надежного выполнения квантовых операций. В отличие от классических компьютеров, где состояния битов являются детерминированными и легко контролируемыми, квантовые компьютеры оперируют кубитами, которые могут находиться в суперпозиции состояний и испытывать квантовые флуктуации. В результате, для обеспечения надежного функционирования квантового процессора необходима тщательная калибровка [1].

Основные задачи калибровки включают в себя настройку временных параметров импульсов управления, балансировку параметров кубитов, компенсацию влияния внешних шумов и дрейфов (изменений оптимумов со временем), а также коррекцию ошибок, возникающих в результате неточностей в квантовых операциях. Этот процесс важен для достижения высокой точности и надежности в вычислениях на квантовом компьютере.

Без калибровки квантовый компьютер может демонстрировать непредсказуемое поведение, что существенно ограничивает его применение в практических приложениях. Недостаточная калибровка может привести к ухудшению качества квантовых операций, увеличению вероятности ошибок и снижению долговечности квантовых систем. Калибровка квантового компьютера является неотъемлемой частью процесса его эксплуатации и развития, обеспечивая оптимальную производительность и надежность в реальных условиях использования.

2. ОПИСАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ

Вычисления на квантовом компьютере осуществляются путем аналоговых операций с большим количеством физических квантовых битов (кубитов). Управляющие сигналы должны быть точно настроены, поскольку

отклонения от идеала приводят к несовершенным амплитудам и фазам, что приводит к ошибкам. Это представляет собой серьезную проблему, поскольку современные аппаратные средства управления и кубитные системы не могут работать одинаково.

Управляющие импульсы каждого кубита должны быть откалиброваны индивидуально, и идеальные параметры управления могут изменяться со временем. Кроме того, для каждого кубита обычно требуется множество различных операций с высокой точностью, таких как вращение одного кубита, многокубитные вентили, измерение и сброс настроек. Этими задачами занимается область оптимального квантового управления, где точное управление квантовыми системами было продемонстрировано в передаче состояний, высокоточных логических операциях, борьбе с дрейфом управляющих параметров и макроскопических квантовых системах. Однако оптимальное квантовое управление обычно применяется к одной операции, такой как калибровка определенного логического элемента или квантового состояния.

Как правило, параметры управления для устройства определяются в соответствии с тщательно подобранной последовательностью калибровочных экспериментов, в ходе которых результаты одного эксперимента, как правило, отражаются на следующем. Простейшая стратегия заключается в последовательном проведении этих экспериментов от начала до конца по порядку. Однако это может быть нарушено из-за изменения параметров, необходимости отладки или желания перенастроить систему на лету. Повторный запуск последовательности с самого начала не является идеальным решением, учитывая значительные временные затраты на калибровку. Необходимо плавно перемещаться как вперед, так и назад по последовательности экспериментов, а также найти способ представить взаимосвязь между экспериментами по калибровке.

Для высокопроизводительных квантовых вычислений требуется система калибровки, которая изучает оптимальные параметры управления намного

быстрее, чем дрейф системы. В некоторых случаях процедура обучения требует решения сложных задач оптимизации, многомерными, с большими ограничениями и огромными пространствами поиска. Такие проблемы препятствуют масштабируемости, поскольку традиционные глобальные оптимизаторы часто слишком неэффективны и медленны даже для небольших процессоров, состоящих из десятков кубитов. Необходимо найти алгоритм, который легко масштабируется в зависимости от количества кубитов и поддается как локальной, так и повторной оптимизации, который можно распараллелить, раскрыв потенциал крупных квантовых процессоров. Рассмотрим несколько существующих подходов к решению этой задачи.

3. ФРЕЙМВОРКИ ОБРАБОТКИ ОШИБОК

3.1. Optimus (Google)

В своей работе [2] Google формализовали проблему калибровки: «Каков оптимальный способ определения и поддержания управляющих параметров для системы физических кубитов при неполной информации о системе?» Для решения они ввели следующий набор терминов [3]:

- Параметр: управляющий параметр кубита, например, длительность импульса pi [4].

- Эксперимент: Набор статических управляющих сигналов и измерений, в которых параметры не изменяются [5]. Это может соответствовать, например, последовательности входов и измерениям для определения распределения вероятностей на выходе или последовательности входов и томографии для определения состояния кубитов. Обычно каждый эксперимент повторяется несколько раз для сбора статистики.

- Сканирование: набор экспериментов, в которых изменяются один или несколько параметров. Например, Rabi Driving [6-7] сканирование, при котором длительность управляющего импульса изменяется для каждого отдельного эксперимента. Сканирование также может быть более сложным, например, многопараметрическая оптимизация.

- Показатель качества: число, измеренное с помощью сканирования или эксперимента, используемое для количественной оценки того, насколько хорошо работает кубит [8].

- Допуск: пороговое значение для определения того, достаточно ли точно определены параметры управления. Например, мы можем указать, что импульс pi должен находиться в пределах 10-4 радиан от π-оборота.

- Спецификация: Показатель качества либо находится в пределах допустимого (согласно спецификации), либо нет (не соответствует спецификации).

Для решения проблемы калибровки, Google разработали алгоритм Optimus. В задумке он должен обладать следующими свойствами:

1. Автоматически калибрует кубитную систему с нуля (из неизвестного исходного состояния).

2. Выбирает последовательность калибровок для выполнения.

3. Занимает минимальное время, затрачиваемое настенными часами.

4. Управляет отклонением параметров, т.е. имеет стратегию для устранения отклонения и повторного выполнения калибровок, которые могли отклониться.

5. Определяет, работает ли калибровка неправильно, с помощью самодиагностики.

Проходить по всей цепи калибровки каждый раз невыгодно (например, если произошел дрифт параметров) [9-10] — необходимо придумать способ двигаться по последовательности экспериментов вперед-назад и показывать связь между ними. Опытный пользователь вручную может двигаться вперед-назад при понимании структуры связей экспериментов. Для создания алгоритма, эксперименты отображаются в виде направленного ациклического графа (DAG) [11]. Представление графа калибровок можно увидеть на рисунке 1. Стрелки зависимостей указывают на корень. Цель калибровки состоит в том, чтобы взять системы с небольшим количеством информации (A) или частичной информацией (B, C) и привести их к состоянию, в котором все вызовы соответствуют спецификации.

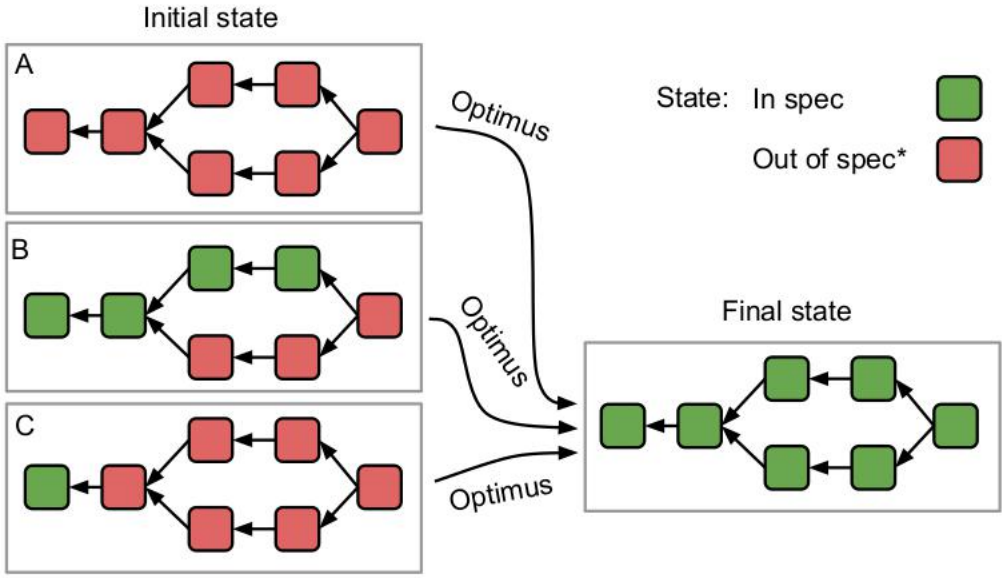

Рис. 1. Граф калибровок Optimus

Для конструктивно разных систем нужны разные графы, даже для одной их может быть несколько. Граф — сбор всех знаний о калибровке системы, пользователи могут менять его и добавлять узлы, если это улучшит результат.

Также необходимо хранить инфу о состоянии системы и журнал запусков каждого узла со статусом выполнения. Калибровка не может быть успешной, если она зависит от неуспешной — это условие играет важную роль при выборе пути обхода. Также, в течение времени, параметры могут изменяться — происходит дрифт. Чтобы бороться с дрифтом параметров, необходимо хранить таймаут для каждой калибровки (промежуток времени, через который ее нужно будет перезапустить).

Optimus состоит из трех концептуальных функций для взаимодействия с калибровками в графе:

1. check_state() — отвечает на вопрос: основываясь на наших предварительных знаниях о системе и без проведения экспериментов, уверены ли мы, что показатели качества, связанные с этой калибровкой, соответствуют спецификации?

Успешна, если одновременно:

- калибровка успешно запускала check_data() или calibrate() в пределах таймаута
- калибровка не сломалась без причин на calibrate()
- зависимости калибровки не перекалибровывались после запуска check_data() или calibrate()
- у всех зависимостей также успешно сработала check_state()

2. check_data() — экспериментально выясняет состояние узла, запуская минимум экспериментов. Проводится минимум экспериментов, результат сравнивается с ожидаемым. Если произошел дрифт — это записывается в state. Если в результате приходит шум — значит сканирование для калибровки не работает — скорее всего проблема в ошибках в его зависимостях.

3. calibrate() — производит измерения для калибровки

Параметры узла калибровки:

- целевой параметр

- сканирования для этого параметра — check_data сканирование (минимум данных) и calibration сканирование (данных уже больше)

- вспомогательные функции для анализа

- погрешности

- таймаут

Для обхода графа созданы две функции: maintain() и diagnose(). Цель функций — минимизировать время на приведение системы в рабочее состояние.

Функция maintain() начинает калибровки на самом близком к корню узлу (минимум зависимостей), который приводит к ошибкам все последующие узлы. Поведение функции отображено на рисунке 2.

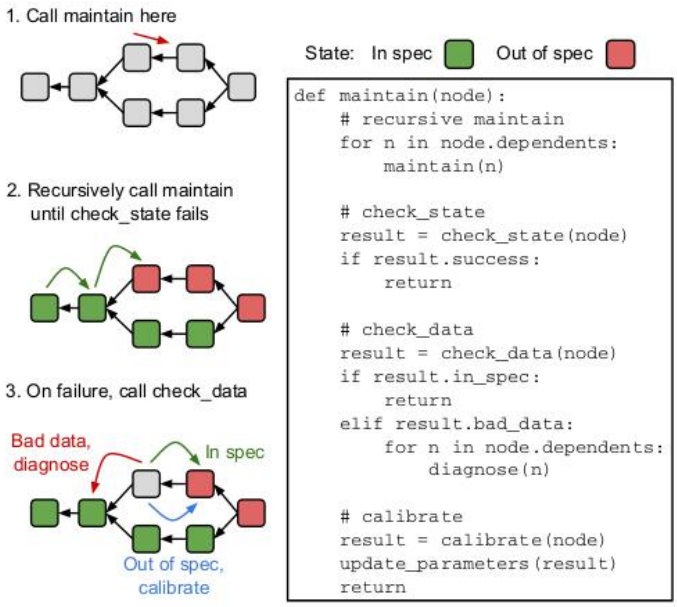

Рис. 2. Алгоритм функции maintain()

Функция diagnose() — вызывается внутри maintain() если check_data показывает что есть bad data (плохие данные). Если в результате bad data, то функция идет дальше по зависимостям, пока не найдет ошибку. Если в результате out of spec (за пределами допуска) — вызывается calibrate(). Поведение функции отображено на рисунке 3.

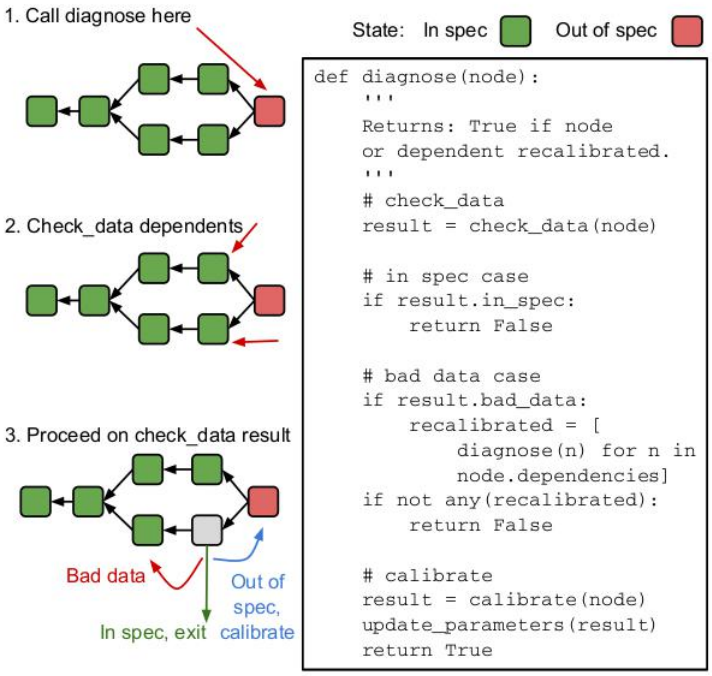

Рис. 3. Алгоритм функции diagnose()

Для оптимизации времени работы важно, что калибровка узлов происходит снизу вверх (чтобы избежать времени на калибровку узла, когда на самом деле узел из его зависимости выдает неправильный результат). Также направление диагностики важно, чтобы найти первый сломанный узел, а не калибровать все подряд от корня.

Если в графе присутствуют циклы — цикл развертывается в цепочку из последовательно повторяющихся узлов, либо необходим эксперимент, где циклические узлы калибруются одновременно.

3.2. Snake Optimizer

Алгоритм предложен командой Google AI Quantum [12], продолжая идею Optimus. Snake Optimizer основан на идеях искусственного интеллекта, оптимизации графов и динамического программирования. Он отображает калиброванные параметры и их взаимодействие на узлах и ребрах графика. Калибровка выполняется путем перемещения по графику с одновременной калибровкой локального подмножества узлов и/или ребер на каждом шаге. Перемещение по процессору визуально напоминает аркадную игру Snake

(отсюда и название). Алгоритм Snake был разработан для оптимизации частот, на которых реализуются квантовые логические элементы в перестраиваемых по частоте сверхпроводящих кубитах. Это позволило команде Google повысить производительность системы на 53-кубитном квантовом процессоре [13], показав потенциал превосходства квантовых вычислений над классическими.

Snake разделен на три компонента (рис. 4). Первый компонент - это стек калибровок [14], в котором каждый уровень представляет один класс калибровок для соответствующих вычислительных элементов. Второй - это система, которая изучает оптимальные параметры соответствующих вычислительных элементов на каждом уровне. Третий компонент - это система, которая перемещается между уровнями калибровки.

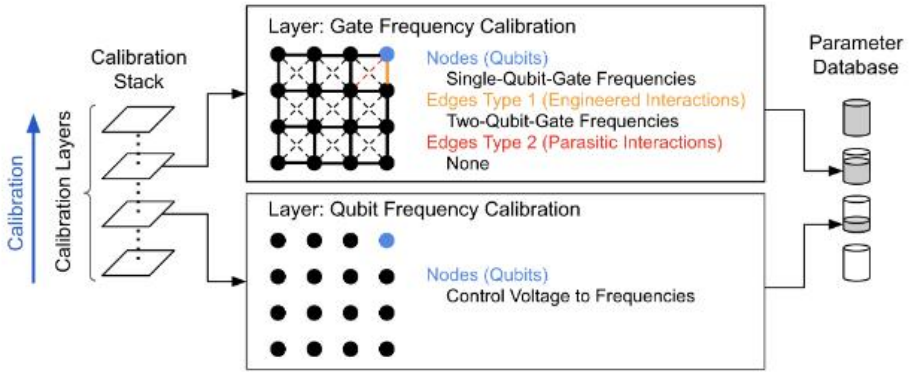

Рис. 4. Обощенная структура калибровочной системы

Ключевым взаимозависимым уровнем калибровки является уровень калибровки частоты квантовых логических элементов [15]. Задача калибровки на этом уровне заключается в настройке всех частот одиночных и двухкубитных элементов в ходе выполнения квантового алгоритма, для которого калибруется процессор. Калибровка взаимозависима, поскольку все кубиты могут взаимодействовать из-за инженерных взаимодействий и/или паразитных перекрестных помех, и, следовательно, все оптимальные частоты зависят друг от друга явно или неявно [16].

Для процессора с N кубитами, связанными с ближайшими соседями, на квадратная решетка, калибровка, таким образом, соответствует решению задачи комбинаторной оптимизации по O (N) измерениям в пространстве поиска $k^{O(N)}$, где k - количество вариантов частоты на элемент. На практике k ~ 10^2, и, таким образом, пространство поиска значительно превышает размерность пространства процессора в 2^N. Учитывая сложность задачи, полный перебор значений является трудноразрешимым, а глобальная оптимизация неэффективна даже для небольших процессоров с десятками кубитов. Алгоритм Snake ищет эффективную стратегию калибровки, которая может быть применена к взаимозависимым калибровочным слоям.

Калибровка выполняется путем обхода графика [17-18] с одновременной калибровкой локального подмножества узлов и/или ребер на каждом шаге. Правила обхода выбираются эвристически, для лучшего достижения цели. Каждый этап калибровки выполняется путем построения и оптимизации модели ошибок для калиброванных узлов и/или ребер в соответствии с ограничениями, налагаемыми ранее откалиброванными узлами и/или ребрами, спецификациями аппаратного обеспечения и квантовым алгоритмом, для которого калибруется процессор. Такой подход является жадной оптимизацией, которая находит локально оптимальные решения на каждом шаге. Применяя динамические ограничения, зависящие от истории калибровки, система эмулирует оптимизацию с более высокой размерностью и возвращает решение, удовлетворяющее всем ограничениям процессора.

Неоткалиброванные элементы разбиваются на подцели алгоритма, которые можно обрабатывать независимо и калибровать

параллельно. Каждая подцель состоит из потока обхода узлов и потока обхода ребер.

На каждом шаге обхода создаются параметры калибровки и активные ограничения калибровки вокруг центрального элемента графика. Затем модель ошибок оптимизируется в соответствии с параметрами калибровки для определения лучших управляющих параметров. Эти параметры сохраняются в базе данных параметров, и калибровка переходит к следующему этапу. На рисунке 5 приведены структуры калибровки для произвольно выбранной цели калибровки G*, состояния калибровки P^* и центрального элемента графика n.

Параметры калибровки P^1_n построены вокруг n с областью $d_P = 1$ (d - расстояние по ребрам). Для этой конфигурации 1 узел и 2 ребра калибруются одновременно, что соответствует трехмерной оптимизации. Активные ограничения калибровки R^2_P с областью $d_R = 2$ для алгоритма XEB (Cross-entropy benchmarking). Эти ограничения создаются на основе активных элементов графика, как показано на нижней части рисунка. Для этой конфигурации параметры калибровки ограничены 5 узлами и 5 ребрами. Аналогичные структуры создаются на каждом шаге обхода процедуры калибровки.

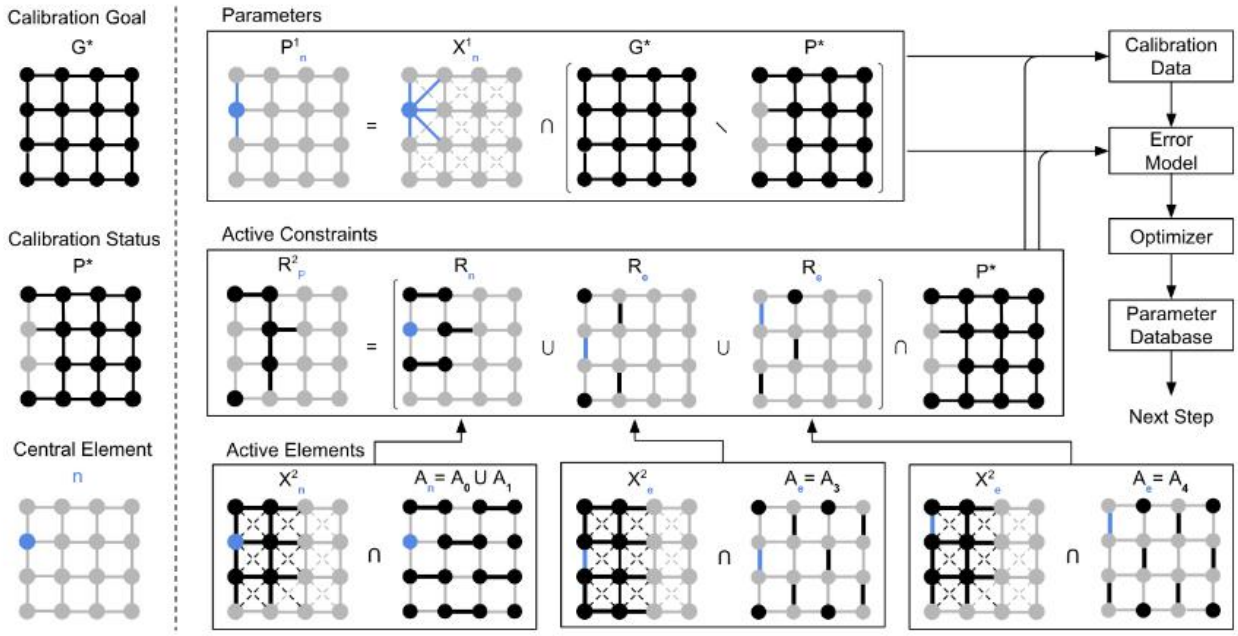

Рис. 5. Операции внутри одного шага калибровки

На рисунке 6 показана последовательность шагов Snake, примененного для оптимизации графа G для калибровки частоты гейта для алгоритма XEB [19]. Неоткалиброванные элементы выделены серым цветом, откалиброванные элементы - черным, параметры калибровки - синим, а активные ограничения калибровки - оранжевым. Для параметра, ограничения и расстояний обхода заданы значения $d_P = 1$, $d_R = 2$, $d_T = 2$ соответственно. Граф заполняется розовым узлом, а обход выполняется с помощью произвольно выбранного правила обхода [20]. Для полной калибровки достаточно одного начального узла, благодаря выбранному правилу обхода и параметру калибровки distance.

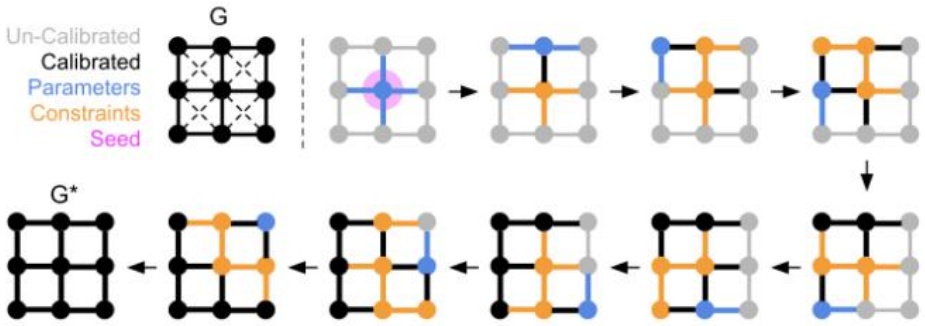

Рис. 6. Последовательность шагов Snake на произвольном графе

4. АНАЛИЗ ПОДХОДОВ

И Snake Optimizer, и Optimus нацелены на эффективное и быстрое решение сложных задач оптимизации в контексте квантовых калибровок. В то время как Snake Optimizer фокусируется на определении оптимальных параметров и взаимодействий на графе и обходе графа калибровки, Optimus формулирует зависимости между калибровками в виде направленного ациклического графа и использует стратегии обхода графика, чтобы минимизировать количество экспериментов, необходимых для калибровки. Optimus позволяет явно представить системные знания в графе удобным способом, давая знания о том, как работает система.

Оба подхода демонстрируют потенциал для оптимизации и эффективной эксплуатации крупномасштабных квантовых процессоров [21]. Но по сути, Snake является уложненным продолжением Optimus. Каждый из алгоритмов делает акцент на разных аспектах последовательности калибровок. Соответственно, при выборе алгоритма для оптимизации процесса калибровок, необходимо учитывать оба подхода, проецируя архитектуру и псевдокод на имеющийся квантовый компьютер. Также, для ускорения вычислений и покрытия всех возможных сценариев калибровки, необходимо переработать опыт инженеров квантового компьютера.

Для однокубитных операций Optimus дает общее представление об архитектуре и последовательности калибровок. При реализации важно будет отработать все возможные сценарии прохождения калибровочного графа, а уже потом переходить к калибровкам внутри одного шага. Граф должен быть динамическим, чтобы его можно было

гибко перестраивать для разных систем. Также, в графе должны быть учтены возможные ошибки. Это позволит графу продолжить работу, не выдавая ошибку и не переходя к управлению инженером.

И в Optimus, и в Snake Optimizer, авторы алгоритмов показывают, что их алгоритмы являются state-of-the-art в своей области, и при правильной реализации дадут значительный прирост в скорости калибровок. А переход от взаимодействий с человеком к автоматизации дает возможность экономить время и создавать более надежные квантовые системы.

5. ЗАКЛЮЧЕНИЕ

В ходе выполнения научно-исследовательской работы на тему создания системы автоматизации калибровки квантового компьютера:

- Произведен анализ предметной области, в ходе которого была раскрыта важность и актуальность создания системы автоматизации калибровки
- Рассмотрены актуальные подходы к решению задач оптимизации калибровочного процесса
- Произведено сравнение двух приведенных алгоритмов и сделаны выводы об их использовании при реализации системы калибровок

В результате работы мы ознакомились с общими принципами, методами и технологиями работы с калибровкой квантовых компьютеров. В результате анализа были сделаны выводы, которые в дальнейшем планируется использовать в ВКРМ.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Орлов, М. А., Нечаев, К. А. / Оценка статистических свойств и криптографической стойкости случайных последовательностей, полученных квантовым компьютером IBM // Безопасность информационных технологий. - 2023 - 30(1) - С.14-26.
2. Kelly, J., O'Malley, P., Neeley, M., Neven, H., & Martinis, J. M. (2018). Physical qubit calibration on a directed acyclic graph. arXiv preprint arXiv:1803.03226.
3. DiVincenzo, David P. "The physical implementation of quantum computation." Fortschritte der Physik: Progress of Physics 48.9-11 (2000): 771-783.
4. Schmidt, P. Oetal, et al. "Spectroscopy using quantum logic." Science 309.5735 (2005): 749-752.
5. Huang, Hsin-Yuan, et al. "Quantum advantage in learning from experiments." Science 376.6598 (2022): 1182-1186.
6. Lisenfeld, Jürgen, et al. "Rabi spectroscopy of a qubit-fluctuator system." Physical Review B 81.10 (2010): 100511.
7. Vijay, R., et al. "Stabilizing Rabi oscillations in a superconducting qubit using quantum feedback." Nature 490.7418 (2012): 77-80.
8. Webber, Mark, et al. "The impact of hardware specifications on reaching quantum advantage in the fault tolerant regime." AVS Quantum Science 4.1 (2022).
9. White, Gregory AL, Charles D. Hill, and Lloyd CL Hollenberg. "Performance optimization for drift-robust fidelity improvement of two-qubit gates." Physical Review Applied 15.1 (2021): 014023.
10. Proctor, Timothy, et al. "Detecting and tracking drift in quantum information processors." Nature communications 11.1 (2020): 5396.
11. Digitale, Jean C., Jeffrey N. Martin, and Medellena Maria Glymour. "Tutorial on directed acyclic graphs." Journal of Clinical Epidemiology 142 (2022): 264-267.
12. Klimov, Paul V., et al. "The snake optimizer for learning quantum processor control parameters." arXiv preprint arXiv:2006.04594 (2020).
13. Arute, Frank, et al. "Quantum supremacy using a programmable superconducting processor." Nature 574.7779 (2019): 505-510.
14. Wittler, Nicolas, et al. "Integrated tool set for control, calibration, and characterization of quantum devices applied to superconducting qubits." Physical review applied 15.3 (2021): 034080.
15. Koch, Jens, et al. "Charge-insensitive qubit design derived from the Cooper pair box." Physical Review A 76.4 (2007): 042319.
16. Li, Fei-Yu, and Li-Jing Jin. "Quantum chip design optimization and automation in superconducting coupler architecture." Quantum Science and Technology 8.4 (2023): 045015.
17. Chhugani, Jatin, et al. "Fast and efficient graph traversal algorithm for cpus: Maximizing single-node efficiency." 2012 IEEE 26th International Parallel and Distributed Processing Symposium. IEEE, 2012.
18. Christofides, Nicos. "The optimum traversal of a graph." Omega 1.6 (1973): 719-732.
19. Boixo, Sergio, et al. "Characterizing quantum supremacy in near-term devices." Nature Physics 14.6 (2018): 595-600.
20. Xu, Ling, and Tony Stentz. "A fast traversal heuristic and optimal algorithm for effective environmental coverage." (2011).
21. Barends, Rami, et al. "Coherent Josephson qubit suitable for scalable quantum integrated circuits." Physical review letters 111.8 (2013): 080502.