МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра Алгоритмической математики

Альтернативный экзамен
по дисциплине «Математическая логика и теория автоматов»
Тема: «Квантовый алгоритм»

Студент гр. 2375 _____ Сардинов М. Г.
Студент гр. 2375 _____ Панов М.Ю.

Санкт-Петербург
2024

Квантовые вычисления и алгоритм HHL (Harrow-Hassidim-Lloyd)

Введение в квантовые компьютеры и квантовые вычисления

Квантовые вычисления представляют собой новую парадигму в области информационных технологий, основанную на принципах квантовой механики. В отличие от классических вычислений, которые основаны на использовании двоичной системы счисления и логических операций над битами (единицами и нулями), квантовые вычисления оперируют с квантовыми системами и используют кубиты(0,1 и в состоянии суперпозиции все промежуточные значения между 0 и 1), которые могут находиться в состоянии суперпозиции и обладать квантовой взаимосвязью между собой — квантовой запутанностью.

Основные принципы квантовой механики, которые лежат в основе квантовых вычислений, включают принцип суперпозиции, принцип запутанности и принцип измерения. Принцип суперпозиции означает, что кубиты могут существовать одновременно во всех возможных состояниях, пока не производится измерение, при котором будет определено их конкретное состояние. Принцип запутанности описывает взаимосвязь между кубитами, когда изменение одного из них немедленно влияет на другие кубиты, вызывая эффекты суперпозиции и интерференции. Принцип измерения позволяет получить определенное значение измеряемой величины кубита, что определяет конечный результат вычисления, однако при измерении состояния система теряет состояние суперпозиции, поэтому на выходе мы можем

получить только одно решение (наша задача состоит в том, чтобы максимизировать шансы верного исхода).

Преимущество квантовых вычислений достигается не за счёт того, что каждая операция становится существенно быстрее, а за счёт того, что общее число необходимых операций для решения определённой задачи сокращается. Квантовые алгоритмы уменьшают сложность решения задачи, сводя экспоненциально трудоёмкий для классического компьютера процесс к полиномиальному количеству операций, что и даёт радикальный выигрыш. Это происходит благодаря способности квантовых систем одновременно обрабатывать множество вариантов решения (суперпозиция) и использовать квантовую интерференцию, чтобы «усилить» правильные из них и «подавить» неправильные, в итоге снижая общее число требуемых вычислительных шагов.

Квантовые вычисления обладают большим потенциалом в решении сложных вычислительных задач, таких как факторизация больших чисел, оптимизация трафика в сетях, моделирование сложных химических и физических систем. Однако, на данный момент, разработка квантовых компьютеров остается сложной задачей, связанной с проблемами квантовой декогеренции и взаимодействиями с окружающей средой.

Полезные для понимания математические понятия и терминология

Прежде чем перейти к основной теме, для лучшего понимания дальнейшего материала стоит кратко и поверхностно напомнить следующие понятия:

Собственные числа – это числа, при подстановке которых в уравнение вида $Av = \lambda v$ (где A – квадратная матрица, а v – ненулевой вектор) вы-

полняется это равенство. Проще говоря, собственные числа – это значения, при которых действие оператора (или матрицы) сводится к умножению собственного вектора на число.

Собственный вектор — это вектор $\mathbf{v}$, который при применении линейного преобразования A изменяет только свою длину (масштабируется) и не меняет направления. Собственный вектор подчиняется уравнению:

$$A\mathbf{v} = \lambda\mathbf{v},$$

где:

- A — квадратная матрица $n \times n$, представляющая линейное преобразование;
- λ — собственное число, соответствующее собственному вектору $\mathbf{v}$;
- $\mathbf{v} \neq \mathbf{0}$ — ненулевой вектор.

Эрмитовы матрицы – это квадратные матрицы, которые совпадают со своим собственным комплексно-сопряжённым транспонированием, то есть $A = A^\dagger$. Их ключевой признак – у них все собственные числа действительные, и они играют важную роль в квантовой механике и теории операторов.

Унитарные операторы – это линейные операторы (или матрицы), которые сохраняют длину и угол между векторами, т.е. являются изометриями. Они удовлетворяют условию $UU^\dagger = U^\dagger U = I$, где $U^\dagger$ – обратная по комплексному спряжению и транспонированию матрица U, а I – единичная матрица.

Векторное пространство кубита представляет из себя двухмерное гильбертово пространство, обозначаемое как C^2 (комплексная плоскость в сте-

пени 2). Каждое из состояний кубита может быть представлено суперпозицией двух базисных состояний, которые обычно обозначаются как 0 и 1. Символ '|' используется для обозначения векторов в гильбертовом пространстве.

Таким образом, состояние кубита может быть описано в виде:

$$\psi = \alpha \left|0\right\rangle + \beta \left|1\right\rangle ,$$

где α и β — комплексные числа, известные как амплитуды. Они определяют вероятности обнаружения кубита в каждом из базисных состояний.

Важно отметить, что сумма квадратов модулей амплитуд должна быть равна единице:

$$|\alpha|^2 + |\beta|^2 = 1.$$

Это условие нормировки гарантирует, что вероятность обнаружения кубита в любом состоянии всегда будет равна 1.

QPE (Quantum Phase Estimation) — это метод в квантовых вычислениях, позволяющий определить «фазу» (числовое значение) некоторой операции над квантовым состоянием.

Для понимания рассмотрим QPE подробнее:

1. Предположим, у нас есть унитарная операция U с собственным состоянием $|\psi\rangle$ и собственным значением вида $e^{2\pi i\varphi}$. Число φ — это и есть искомая фаза ($0 \leq \varphi < 1$).

2. Мы готовим несколько «счетных» кубитов в состоянии суперпозиции и связываем их с применением U к состоянию $|\psi\rangle$ в степенях 2^k: $U^{2^0}, U^{2^1}, U^{2^2}$ и так далее.

3. Каждое такое применение «кратно» умножает фазу φ во входной суперпозиции, в итоге создавая состояние, в котором фаза φ зашифрована

в виде степеней двойки.

4. Затем мы применяем обратное квантовое преобразование Фурье (QFT^{-1}), которое «распаковывает» эти фазы в двоичное представление числа φ, после чего измерение даёт приближение фазы в виде обычного числа.

Постановка задачи, проблематика классического решения и описание ее в квантовой форме.

Официальная постановка задачи:

Ускорение: Суперполиномиальное

Реализация: Classiq(hhl), Cirq(hhl)

Описание: Нам предоставлен доступ к оракулу $n \times n$ матрицы A и некоторое описание вектора b. Мы хотим найти некоторое свойство $f(A)b$ для некоторой эффективно вычисляемой функции f. Предположим, A — эрмитова матрица с $O(\text{polylog } n)$ ненулевыми элементами в каждой строке и числом обусловленности k. Как показано в литературе, квантовый компьютер может за время $O(k^2 \log n)$ вычислять с полиномиальной точностью различные ожидаемые значения операторов по отношению к вектору $f(A)b$ (при условии, что квантовое состояние, пропорциональное b, эффективно создаваемо). Для некоторых функций, таких как $f(x) = 1/x$, этот метод можно обобщить на неэрмитовы и даже неквадратные A. Время выполнения этого алгоритма впоследствии было улучшено до $O(k \log^3 k \log n)$.

Экспоненциальное улучшение масштабирования с точностью было достигнуто в последующих исследованиях. Были предложены некоторые методы для расширения этого алгоритма на неразреженные матрицы, хотя

они требуют предварительного вычисления определённых частичных сумм элементов матрицы.

Некоторые ограничения квантового обучения, основанного на линейных системах, хорошо описаны в литературе. Было также продемонстрировано, что квантовые компьютеры могут решать хорошо обусловленные $n \times n$ матрицы, используя всего $O(\log n)$ кубитов, в то время как лучшие классические алгоритмы используют $O(\log^2 n)$ битов.

Варианты задачи линейных систем, включая вычисление псевдообратных матриц, могут быть получены как частные случаи квантового преобразования сингулярных значений.

Классический подход

Классическое решение задачи линейных систем уравнений $Ax = b$ включает нахождение вектора x для заданной матрицы A и вектора b. Наиболее распространённые методы решения включают:

1. *Методы прямого решения:* Такие методы, как метод Гаусса и разложение LU, представляют собой эффективные алгоритмы для решения систем с плотными матрицами. Их временная сложность составляет $O(n^3)$ для матриц размером $n \times n$, если не учитывать специальные структуры.

2. *Итеративные методы:* Алгоритмы, такие как метод сопряжённых градиентов, предназначены для решения разреженных и плохо обусловленных систем. Эти методы не вычисляют точное решение напрямую, но приближаются к нему, что делает их более подходящими для больших систем. Их временная сложность может быть существенно ниже, особенно если матрица имеет разреженную структуру.

3. *Использование псевдообратной матрицы:* Для плохо обусловленных или неквадратных систем можно использовать матрицу Мура–Пенроуза, однако этот метод может быть вычислительно затратным.

Классические подходы обеспечивают высокую точность, но сталкиваются с проблемами при обработке больших данных или разреженных матриц, где ресурсы (время и память) становятся ограничивающими факторами.

Квантовый подход: Алгоритм HHL

Квантовый алгоритм HHL предлагает альтернативный подход для решения линейных систем. Основные этапы работы квантового алгоритма можно описать следующим образом:

1. *Квантовая оценка фазы (QPE):* Входные данные преобразуются в квантовое состояние, представляющее матрицу A и вектор b. На этом этапе вычисляются собственные значения матрицы A и их представление в двоичной форме.

2. *Инверсия собственных значений:* На основе вычисленных собственных значений выполняется их инверсия, что соответствует решению системы $Ax = b$ в собственном базисе.

3. *Обратная оценка фазы:* Применяется обратное преобразование, которое возвращает квантовую систему в исходный базис, предоставляя состояние, пропорциональное решению x.

4. *Измерение вспомогательного кубита:* Если результат измерения вспомогательного кубита равен 1, система готова представить нормализованное решение задачи.

Предисловие к HHL.

Алгоритм HHL (Harrow-Hassidim-Lloyd) представляет собой квантовый алгоритм, предназначенный для решения линейных систем уравнений. С момента его появления в 2009 году он стал одним из наиболее значимых достижений в области квантовых вычислений благодаря своей способности демонстрировать ускорение по сравнению с классическими методами для определённых типов задач.

В классической математике задача решения линейной системы уравнений $Ax = b$ заключается в нахождении вектора x, где A — известная матрица, а b — известный вектор. Однако в случае больших и сложных систем вычисления могут стать крайне трудоёмкими. Квантовый алгоритм HHL предоставляет способ получить квантовое представление решения линейной системы значительно быстрее, чем это возможно с использованием классических методов, при условии, что матрица A удовлетворяет определённым ограничениям, таким как разреженность, эрмитовость и хорошая обусловленность.

Основная идея алгоритма HHL заключается в преобразовании задачи решения линейной системы в набор квантовых операций, включая квантовую оценку фазы, преобразования собственных значений и их инверсию. Итогом работы алгоритма является состояние, пропорциональное вектору решения, закодированное в квантовом регистре.

Алгоритм HHL нашёл применение в различных областях, включая машинное обучение, финансовый анализ, обработку сигналов и моделирование физических систем. Несмотря на свои ограничения, такие как необходимость дополнительных ресурсов для подготовки входных данных и сложности с декодированием результата, HHL служит важным примером

того, как квантовые компьютеры могут превосходить классические методы в определённых вычислительных задачах.

Дальнейшее описание алгоритма и его шагов приведено ниже. На высоком уровне алгоритм состоит из трёх основных шагов: оценка фазы, вращение для инверсии собственных значений и обратная оценка фазы.

Процедура алгоритма выглядит следующим образом:

1. **Применение квантовой оценки фазы (QPE):**

   После выполнения QPE регистр, выраженный в базисе собственных векторов матрицы A, записывается как:

   $$\sum_j \beta_j \left|u_j\right\rangle \left|\tilde{\lambda}_j\right\rangle,$$

   где n_l представляет количество бит для двоичного представления каждого собственного значения, а $n_b = \log(N)$ — количество кубитов, представляющих $|b\rangle$.

2. **Применение вращения вспомогательного кубита:**

   Применяется вращение вспомогательного кубита, чтобы получить нормализованное состояние вида:

   $$\sum_j \beta_j \left|u_j\right\rangle \left|\tilde{\lambda}_j\right\rangle \left(\sqrt{1 - \frac{C^2}{\lambda_j^2}}\left|0\right\rangle + \frac{C}{\lambda_j}\left|1\right\rangle\right),$$

   где эти вращения зависят от каждого собственного значения матрицы A. Они могут быть выполнены с использованием матрицы вращения Y, которая задаётся как:

   $$R_y(\theta) = \begin{bmatrix} \cos(\theta/2) & -\sin(\theta/2) \\ \sin(\theta/2) & \cos(\theta/2) \end{bmatrix}.$$

Здесь C является масштабным коэффициентом, используемым для нормализации квантовых состояний.

3. **Обратная оценка фазы:**

   На последнем шаге алгоритма выполняется обратная квантовая оценка фазы. Это приводит регистр часов обратно в состояние $|0\rangle$ и оставляет оставшееся состояние как:

$$\sum_j \beta_j \frac{C}{\lambda_j} |u_j\rangle |0\rangle .$$

   Если вспомогательный кубит находится в состоянии $|1\rangle$, то состояние регистров является нормализованным эквивалентом вектора решения. Поэтому измеряется вспомогательный кубит, и если результат равен 1, то новое состояние принимает вид:

$$|x\rangle = \sum_j \beta_j \frac{1}{\lambda_j} |u_j\rangle ,$$

   которое пропорционально компонентам вектора решения $|x\rangle$.

Математическое обоснование корректности работы алгоритма

Рассмотрим уравнение $A\vec{x} = \vec{b}$, предполагается, что A эрмитова, следовательно, можно представить матрицу A в следующем виде:

$$A = \sum_{j=0}^{N-1} \lambda_j |u_j\rangle\langle u_j|$$

Это разбивает матрицу A на собственные значения (λ_j) и собственные векторы ($|u_j\rangle$). Разложение говорит, что A можно представить как сумму вкладов от всех собственных векторов $|u_j\rangle$, причём каждый вклад масштабируется соответствующим собственным значением λ_j. Тогда можно представить A^{-1} в соответствующем виде.

$$A^{-1} = \sum_{j=0}^{N-1} \lambda_j^{-1} |u_j\rangle\langle u_j|$$

Поскольку A обратима и эрмитова, она должна иметь ортогональный базис собственных векторов, и поэтому мы можем записать b как

$$|b\rangle = \sum_{j=0}^{N-1} b_j |u_j\rangle, \quad b_j \in \mathbb{C}$$

Перейдем к рассмотрению непосредственно алгоритма. Начнем с трех регистров: вспомогательного регистра S, регистра часов C и входного регистра I. Для начала необходимо использовать QPE для нахождения собственных значений матрицы A :

$$U = e^{iAt} = \sum_{j=0}^{N-1} e^{i\lambda_j t} |u_j\rangle\langle u_j|$$

Теперь регистр, выраженный в собственном базисе A, теперь равен:

$$\sum_{j=0}^{N-1} b_j |\lambda_j\rangle_{nl} |u_j\rangle_{nb}.$$

$|\lambda_j\rangle_{nl}$: Это квантовая запись собственного значения λ_j, полученная в процессе фазовой оценки. - nl — количество бит для бинарного представления λ_j. Например, если точность равна 2^{nl}, то можно представить λ_j с такой точностью. Это нужно для точного представления фазовых соб-

ственных значений.

$|u_j\rangle_{nb}$: Собственный вектор $|u_j\rangle$, представленный в регистре из nb кубитов. - $nb = \log(N)$, где N — размерность пространства (число базисных векторов).

Теперь к этому состоянию добавляют вспомогательный кубит и применяют вращение для нормализации, в результате чего состояние принимает вид:

$$\sum_{j=0}^{N-1} b_j |\lambda_j\rangle_{nl} |u_j\rangle_{nb} \left(\sqrt{1 - \frac{C^2}{\lambda_j^2}} |0\rangle + \frac{C}{\lambda_j} |1\rangle \right),$$

- $|0\rangle$ и $|1\rangle$: состояния вспомогательного кубита. - C: масштабирующий фактор, который обеспечивает нормализацию. Эти вращения обусловлены каждым собственным значением. Они могут быть достигнуты путем применения матрицы вращения Y, где

$$\theta = \cos^{-1}\left(\frac{C}{\lambda_j}\right)$$

Также обратим внимание, что здесь C — это коэффициент масштабирования, который мы используем, чтобы гарантировать, что все наши квантовые состояния нормализованы.

Последним шагом алгоритма является выполнение обратной оценки квантовой фазы. Это устанавливает регистр часов обратно в 0 и оставляет оставшееся состояние как

$$\sum_{j=0}^{N-1} b_j |0\rangle_{nl} |u_j\rangle_{nb} \left(\sqrt{1 - \frac{C^2}{\lambda_j^2}} |0\rangle + \frac{C}{\lambda_j} |1\rangle \right).$$

Обратим внимание: если вспомогательный кубит равен 1, то состояние регистров является нормализованным эквивалентом вектора решения.

Итак, мы измеряем этот вспомогательный кубит, и если мы действительно получим 1, новое состояние будет

$$\frac{1}{\sqrt{\sum_{j=0}^{N-1} \frac{|b_j|^2}{\lambda_j^2}}} \sum_{j=0}^{N-1} \frac{b_j}{\lambda_j} |u_j\rangle_{nb}.$$

Поэтапное рассмотрение кода, и построение схемы по нему

Шаг 1: Инициализация регистров и схемы

Сначала создаются квантовые регистры:

- b_reg: регистр для кодирования вектора $|b\rangle$ размером 2 кубита.
- $phase\_reg$: фазовый регистр для выполнения фазовой оценки размером 3 кубита.
- anc_reg: вспомогательный кубит для контроля операций.
- c_reg: классический регистр для сохранения результатов измерений.

```
from qiskit import QuantumCircuit, QuantumRegister, ClassicalRegister
from qiskit.extensions import UnitaryGate
import numpy as np

num_qubits_for_b = 2
num_qubits_for_phase = 3
ancilla = 1
```

```
total_qubits = num_qubits_for_b + num_qubits_for_phase + ancilla

b_reg = QuantumRegister(num_qubits_for_b, name='b')
phase_reg = QuantumRegister(num_qubits_for_phase, name='phase')
anc_reg = QuantumRegister(ancilla, name='anc')
c_reg = ClassicalRegister(1, name='c')

qc = QuantumCircuit(b_reg, phase_reg, anc_reg, c_reg)
```

—

Шаг 2: Инициализация вектора $|b\rangle$

Задаётся начальное состояние для $|b\rangle$ как $|01\rangle$, что соответствует вектору $b = [0, 1, 0, 0]$.

```
qc.x(b_reg[0])
```

—

Шаг 3: Кодирование матрицы A

Для представления матрицы A используется унитарный оператор U. В данной реализации оператор U задаётся как единичная матрица. В реальной задаче U соответствует матрице, связанной с собственными значениями λ.

```
U_matrix = np.eye(4)
U_gate = UnitaryGate(U_matrix)
```

Шаг 4: Прямая фазовая оценка (QPE)

На данном этапе используется фазовый регистр для выполнения фазовой оценки. Гейты H создают суперпозицию, а контролируемые применения оператора U связывают фазовый регистр с регистром b_reg, что позволяет выполнить фазовую оценку для различных собственных значений.

```
for i in range(num_qubits_for_phase):
    qc.h(phase_reg[i])
    qc.append(U_gate.control(1), [phase_reg[i]] + b_reg[:])
```

—

Шаг 5: Ротация вспомогательного кубита

Вспомогательный кубит *ancilla* подвергается вращению на угол θ, зависящий от собственных значений матрицы A. Управляющими кубитами для этой операции являются кубиты фазового регистра.

```
theta = np.pi/4
control_qubits = phase_reg[:]
qc.mcry(theta, control_qubits, anc_reg[0])
```

—

Шаг 6: Обратная фазовая оценка (Inverse QPE)

Для восстановления фазового регистра в исходное состояние выполняется обратная фазовая оценка. Обратные операции включают:

- Применение обратного контролируемого оператора U^{2^i}.
- Применение обратных гейтов Адамара H^{-1}.

```
for i in reversed(range(num_qubits_for_phase)):
    qc.append(U_gate.control(1).inverse(), [phase_reg[i]] + b_reg[:])
    qc.h(phase_reg[i])
```

—

Шаг 7: Измерение вспомогательного кубита

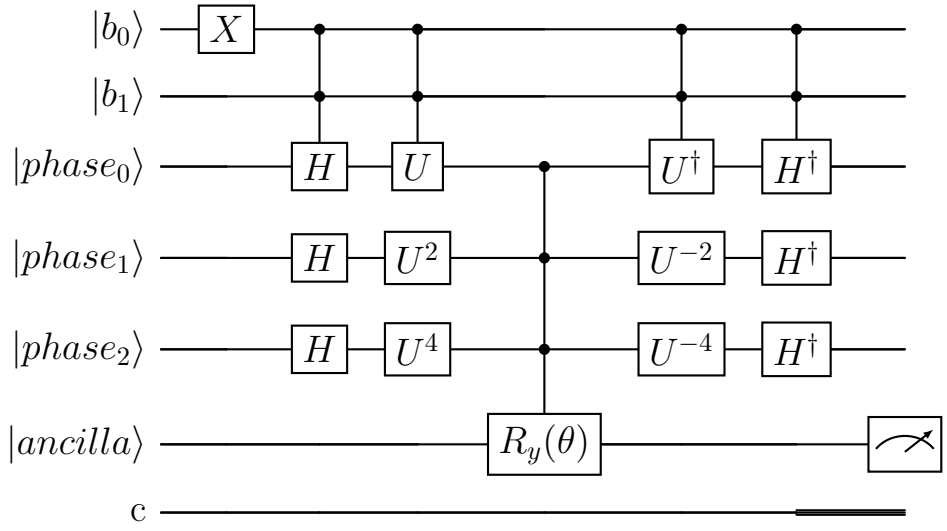
На последнем шаге выполняется измерение вспомогательного кубита $ancilla$, результат которого сохраняется в классический регистр c_reg.

```
qc.measure(anc_reg[0], c_reg[0])
```

—

Итоговая квантовая схема

В результате всех описанных шагов формируется квантовая схема, отображающая алгоритм решения задачи линейных систем.



Оценка временной сложности алгоритма HHL и классических алгоритмов

Классические методы, такие как метод Гаусса, LU-разложение или итеративные алгоритмы (например, метод сопряжённых градиентов), имеют временную сложность, зависящую от размера матрицы n. Для плотных матриц она составляет $O(n^3)$, а для разреженных может быть снижена до $O(n \log n)$ или $O(n)$ в некоторых случаях. Тем не менее, эти алгоритмы ограничены линейным масштабом времени в зависимости от числа обусловленности κ, что делает их менее эффективными для сильно масштабированных задач.

С другой стороны, алгоритм HHL имеет временную сложность $O(\log n)$, которая экспоненциально меньше по сравнению с классическими методами, но это достижимо только при выполнении следующих условий:

1. матрица должна быть разреженной (с числом ненулевых элементов в каждой строке, полиномиальным относительно n);
2. подготовка состояния $|b\rangle$ должна быть эффективной;
3. число обусловленности κ должно быть полиномиальным.

В реальных сценариях, особенно при больших числах обусловленности κ, квантовое преимущество может быть утрачено. Более того, HHL даёт квантовое состояние $|x\rangle$, что требует дополнительных усилий для извлечения классического результата, тогда как классические алгоритмы непосредственно предоставляют вектор $\mathbf{x}$.

Заключение

Таким образом, алгоритм HHL демонстрирует важность квантовых вычислений для решения фундаментальных задач линейной алгебры. Хотя его применение в реальных условиях требует преодоления значительных технических ограничений, способность обеспечивать экспоненциальное ускорение делает его важным инструментом для будущих исследований в области квантовых алгоритмов. Однако для задач с низкой размерностью или плохо обусловленными матрицами классические подходы пока остаются более практичными. Дальнейшее развитие квантовых технологий и оптимизация алгоритма HHL могут расширить его применение в реальных задачах, приближая его к широкому использованию в науке и промышленности.

Ссылки на литературу (и просто полезная инфа)

https://www.sciencedirect.com/science/article/abs/pii/S037596012030462X
https://iopscience.iop.org/article/10.1088/1742-6596/2333/1/012023
https://ieeexplore.ieee.org/abstract/document/10189828