УДК 004.021

Алгоритм HHL: обзор, принцип работы, примеры использования

Д.С. Амирханов
Научный руководитель: к.ф.-м.н. Б.С. Мерзликин
Национальный исследовательский Томский политехнический университет,
Россия, г. Томск, пр. Ленина, 30, 634050
E-mail: dsa30@tpu.ru

HHL algorithm: overview, operating principle, examples of use

D.S. Amirkhanov
Scientific Supervisor: Ph.D. B.S. Merzlikin
Tomsk Polytechnic University, Russia, Tomsk, Lenin str., 30, 634050
E-mail: dsa30@tpu.ru

Abstract. *We consider a short overview of the quantum algorithm Harrow-Hasidim-Lloyd (HHL) for solving system of linear equations. We provide a mathematical formulation of the algorithm, the corresponding quantum circuit, and discuss examples of application the algorithm to solve a system of linear equations. We implement the algorithm under consideration using a simulation quantum computer, the results of simulation we compare with the classical solution algorithm for linear systems.*

Key words*: Quantum computing, qubit, computational complexity, solutions of systems of linear equations.*

Введение

Системы линейных алгебраически уравнений (СЛАУ) играют огромную роль в науке и часто встречаются в физике, биологии, экономике, инженерных задачах. Классические способы решения СЛАУ размера $N \times N$, например, методы Гаусса и Крамера, требуют полиномиального времени для решения, то есть имеют вычислительную сложность $O(n^3)$ и $O(n^4)$ соответственно. В случае больших систем уравнений получить решение за разумное время не представляется возможным: к таким задачам можно отнести, например, моделирования потока в трещинах горной породы [1] или алгоритмы машинного обучения [2]. В таких ситуациях полностью раскрывается преимущество квантовых вычислительных алгоритмов перед классическими, поскольку они способны обеспечить экспоненциальное ускорение при решении некоторых задач. К таким алгоритмам относится алгоритм Харроу-Хассидим-Ллойда (HHL), который, при условии разряженной СЛАУ с числом обусловленности k, имеет вычислительную сложность $O(\log(N)\, k^2)$, где N – число переменных в СЛАУ [3].

Цель работы – изучение работы алгоритма HHL и его моделирование на классическом компьютере.

Экспериментальная часть

СЛАУ может быть представлена следующим образом:

$$A\vec{x} = \vec{b}, \tag{1}$$

где A – матрица размера $N \times N$, $\vec{x}$ и $\vec{b}$ – вектора размерности N. Для простоты предполагается, что $N = 2^n$, где n – число кубит в квантовой схеме. $\vec{x}$ – неизвестный вектор, A и $\vec{b}$ известны. Тогда, вектор $\vec{x}$ может быть найдет просто как:

$$\vec{x} = A^{-1}\vec{b} \tag{2}$$

Предполагается, что матрица A эрмитова. В случае квантовых алгоритмов (1) может быть переписано в нотациях Дирака, а именно $A|x\rangle = |\mathrm{b}\rangle$, тогда (2) запишется как $|x\rangle = A^{-1}|b\rangle$.

Далее кратко рассмотрим математические основы алгоритма HHL, на рис. 1 приведена схема данного квантового алгоритма.

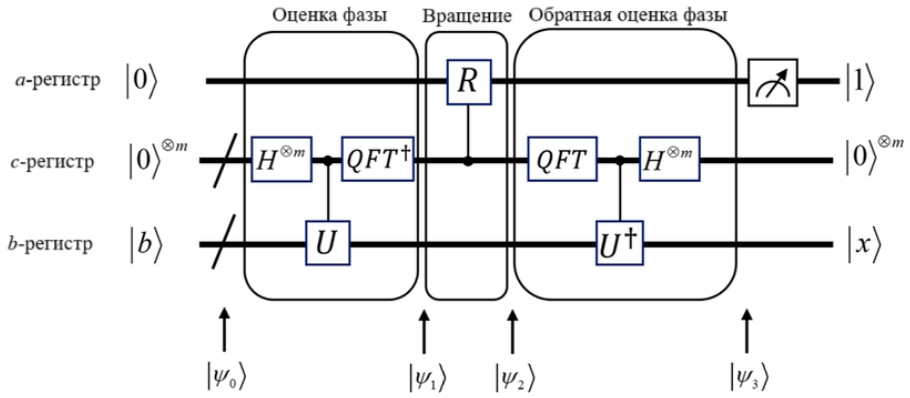

Рис. 1. Схема квантового алгоритма HHL [4]

Первым этапом алгоритма является приготовление состояния, в результате чего состояние системы будет:

$$|\Psi_0\rangle = |0\rangle|0\rangle^{\otimes m}|b\rangle \quad (3)$$

Вторым идет модуль алгоритма квантовой оценки фазы (*QPE*): тактовые кубиты переводятся в состоянии суперпозиции при помощи гейта Адамара H, затем к кубитам b-регистра применяется оператор управляемого вращения U, кубиты c-регистра являются контрольными. Последней операцией в рамках *QPE* являются операция обратного квантового преобразования Фурье (*IQFT* или $QFT^{\dagger}$), применяемого к кубитам c-регистра, в результате чего состояние системы после *QPE* запишется следующим образом:

$$|\Psi_1\rangle = \sum_{j=1}^{N} \beta_j \, |\phi_j\rangle|u_j\rangle|0\rangle \quad (4)$$

Следующий этап: операция управляемого вращения R, переводящая систему в следующее состояние:

$$|\Psi_2\rangle = \sum_{j=1}^{N} \beta_j \left(\sqrt{1 - {\mathrm{C}^2}/{\phi_j^2}}\,|0\rangle + {C}/{\phi_j}\,|1\rangle \right) |\phi_j\rangle|u_j\rangle|0\rangle \quad (5)$$

Последним блоком является блок обратной квантовой оценки фазы, который переводит систему в состояние:

$$|\Psi_3\rangle = \sum_{j=1}^{N} \beta_j \left(\sqrt{1 - {\mathrm{C}^2}/{\phi_j^2}}\,|0\rangle + {C}/{\phi_j}\,|1\rangle \right) |0\rangle^{\otimes m}|u_j\rangle|0\rangle \quad (6)$$

Как итог, в том случае, если кубит из a-регистра измеряется как $|1\rangle$, кубиты b-регистра коллапсируют в $|x\rangle = C \sum_{j=1}^{N} \left({\beta_j}/{\phi_j}\right)|u_j\rangle$. Таким образом, получаем исходный вектор $\vec{x}$, определяемый с точностью до нормировочной константы C [4, 5].

Моделирование алгоритма будет проводится на классическом компьютере на языке программирования Python, с использованием библиотеки для имплементации квантовых вычислений Qiskit. Для получения классического решения использовалась библиотека Scipy. В целях тестирования была взята следующая система линейных уравнений:

$$\begin{gathered} 2x_1 + (1+i)x_2 + (9+2i)x_3 = 4+i \\ (1-i)x_1 + 5x_2 + (4+6i)x_3 + (3-2i)x_4 = 1 \\ (9-2i)x_1 + (4-6i)x_2 + 10x_3 + (1+7i)x_4 = -i \\ (3+2i)x_2 + (1-7i)x_3 + 4x_4 = 7+3i \end{gathered} \quad (*)$$

Как было указано выше, систему уравнений в матричной форме можно представить как (1). Необходимо найти решение в виде вектора (2).

Результаты

Рассмотрим результаты работы классического алгоритма (с точностью до тысячных):

$$\vec{x}_c = (1.863 - 1.441i, \quad -0.633 - 0.482i, \quad 0.154 - 0.514i, \quad 1.033 + 1.588i)$$

Подставляя полученное решение в (*), получим тождество. Далее рассмотрим результаты работы алгоритма HHL. В этом случае получим следующий результат:

$$\vec{x}_{hhl} = (0.320 - 0.248i, \quad -0.108 - 0.083i, \quad 0.027 - 0.088i, \quad 0.118 + 0.059i)$$

На первый взгляд кажется, что результаты работы алгоритма HHL абсолютно не сходятся с решением, которое нашел классический алгоритм, и подстановка в (*) тождества не дает. Но данный алгоритм не находит точное решение СЛАУ, а возвращает *функцию* вектора решения, которая, в общем случае, связана с самим вектором решения через константу. Для того, чтобы избавиться от влияния констант на выходное решение, вектор $\vec{x}_{hhl}$ на соответствующую ему норму, после чего умножим полученный вектор на соответствующую ему евклидову норму *полного* вектора решения, тогда получим:

$$\vec{x}_{hhl} = (1.866 - 1.442i, \quad -0.631 - 0.481i, \quad 0.154 - 0.514i, \quad 1.033 + 1.586i)$$

Видим, что результаты практически одинаковы. Также можем оценить количество гейтов в алгоритме для решения СЛАУ $n \times n$. Результаты оценки указаны в табл. 1.

Таблица 1

Количество квантовых гейтов, необходимых для СЛАУ различного размера

Размер СЛАУ	2 × 2	4 × 4	8 × 8	16 × 16
Количество квантовых гейтов, необходимых для реализации HHL	334	2593	34008	403899

Заключение

В результате работы был изучен алгоритм HHL для решения СЛАУ на квантовом компьютере. Была проведена имплементация данного алгоритма на классическом компьютере, проведено сравнение результата работы классического алгоритма СЛАУ с работой имплементации HHL. Также важно отметить, что на существующих квантовых компьютерах невозможно реализовать алгоритм HHL, поскольку он требует большого количества квантовых гейтов для реализации.

Список литературы

1. Henderson J.M. и др. Quantum algorithms for geologic fracture networks // Scientific Reports. – 2023. – № 1 (13). – P. 2906.

2. Aggarwal C.C., AggarwalL F. Lagerstrom-Fife. Linear algebra and optimization for machine learning. – Cham : Springer International Publishing, 2020. – P. 156.

3. Ambainis A. Variable time amplitude amplification and quantum algorithms for linear algebra problems // 29th International Symposium on Theoretical Aspects of Computer Science (STACS 2012), February 29th - March 3rd, 2012, Paris, France, 2012. – P. 636–647.

4. Gao F. et al. Hybrid algorithms to solve linear systems of equations with limited qubit resources // Quantum Information Processing. – 2022. – № 3 (21). – P. 111.

5. Jr H.J.M., Zaman A., Wong H.Y. Step-by-Step HHL Algorithm Walkthrough to Enhance the Understanding of Critical Quantum Computing Concepts // IEEE Access. – 2023. – Vol. 11. – P. 77117–77131.