Московский государственный университет имени М.В. Ломоносова
Факультет вычислительной математики и кибернетики
Кафедра суперкомпьютеров и квантовой информатики.

Васильев Семён Михайлович

Исследование эффективности применения эволюционных алгоритмов, основанных на квантовом формализме, для настройки гиперпараметров моделей машинного обучения

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

Научный руководитель:
к.ф.-м.н., доцент
Н. Н. Попова

Москва, 2021

Оглавление

1. Введение.....3
1.2 Описание классического генетического алгоритма. 4
1.3 Понятие квантового кубита. 7
1.4 Модель генетического алгоритма, основанного на квантовом формализме. 7
1.5 Цель работы.....10
1.6 Постановка задачи. 10
2. Обзор работ по применению квантового генетического алгоритма для решения задач оптимизации и подбора архитектур нейронных сетей. 10
3. Разработка квантового генетического алгоритма для настройки гиперпараметров сверточных нейронных сетей. 19
3.1 Библиотека функций для построения генетических алгоритмов. 20
3.1.1 Классический генетический алгоритм: 20
3.1.2 Квантовый генетический алгоритм: 21
3.2 Параллельные реализации классического и квантового генетических алгоритмов. 22
3.2.1 Параллельный классический генетический алгоритм. Предлагаемая реализация операции отбора. 23
3.2.2 Параллельный классический генетический алгоритм. Предлагаемая реализация операции скрещивания. 25
3.2.3 Параллельный квантовый генетический алгоритм. Предлагаемая реализация операции обновления лучшего вектора-наблюдения. 26
3.3 Метод настройки гиперпараметров сверточной нейронной сети с использованием ГА и КГА. 27
3.4 Программная реализация процесса конструирования и обучения нейронных сетей. 29
4. Вычислительный эксперимент. 32
4.1 Набор данных. 32
4.2 Алгоритмы и их конфигурации. 32
4.3 Аппаратная и программная платформы, использованные в вычислительном эксперименте. 33
5. Результаты вычислительного эксперимента. 34
6. Заключение. 36
7. Возможные пути дальнейших исследований: 36
8. Список литературы. 37

1. Введение.

1.1 Общая характеристика алгоритмов оптимизации гиперпараметров моделей машинного обучения.

При проектировании любой системы машинного обучения неизбежно встает задача подбора конфигурации её гиперпараметров, на которой система будет показывать наилучшее качество.

Гиперпараметры в машинном обучении - это параметры, значения которых используется для управления процессом обучения алгоритма. Эти значения устанавливаются перед запуском процесса обучения. В этом смысле они и отличаются от обычных параметров, определяемых непосредственно в процессе обучения.

Задача поиска эффективного алгоритма оптимизации гиперпараметров моделей машинного обучения является актуальной по следующим причинам:

- активным использованием алгоритмов машинного обучения при решении современных научных и прикладных задач;
- существенным влиянием гиперпараметров на качество системы машинного обучения;
- высокой вычислительной сложностью процесса обучения моделей машинного обучения;
- использованием параллельных вычислительных систем для сокращения времени, необходимого для подбора оптимальной конфигурации гиперпараметров.

На данный момент популярными являются следующие подходы к решению данной задачи.

- Поиск по решётке. Данный подход заключается в полном переборе по заданному подмножеству пространства поиска. При вещественном или неограниченном диапазоне значений некоторого параметра необходимо вручную назначать границы поиска и дискретизацию. Плюсы алгоритма: простая реализации и масштабируемость. При этом явными недостатками являются экспоненциальный рост сложности в зависимости от количества гиперпараметров и отсутствие учёта истории уже просмотренных точек пространства поиска.

- Случайный поиск. Вместо полного перебора точек заданной сетки в этом подходе производится их отбор случайным образом из пространства поиска. Алгоритмы случайного поиска хорошо масштабируются, но в отличие от поиска по решётке они избавлены от человеческого фактора в виде ручной настройки параметров. Особенно хорошо этот подход показывает себя в сравнении с полным перебором в ситуациях, когда качество модели зависит значительно только от небольшого подмножества своих параметров.
- Байесовская оптимизация. Алгоритмы, основанные на байесовской оптимизации, состоят из двух компонентов: вероятностной модели, называемой суррогатной, и функции выбора следующей точки. Суррогатная модель является дешёвой с вычислительной точки зрения аппроксимацией целевой функции. Она перестраивается на каждой итерации на основании всей истории вычисленных ранее значений целевой функции. После чего функция выбора точки, используя текущее распределение суррогатной модели, отбирает очередную конфигурацию гиперпараметров, балансируя между использованием уже имеющейся информации и исследованием новых областей пространства поиска.
- Эволюционная оптимизация. Этот подход реализует методологию решения задачи глобальной оптимизации. С использованием этого подхода была спроектирована целая группа алгоритмов, получившая название эволюционные алгоритмы. К этой группе алгоритмов относятся генетический алгоритм, алгоритм муравьиной колонии, алгоритм роя частиц, алгоритм дифференциальной эволюции и другие. Операции, используемые в данных алгоритмах, вдохновлены процессами, происходящими в живой природе.

1.2 Описание классического генетического алгоритма.

Генетический алгоритм является эвристическим алгоритмом, предназначенным для решения задачи оптимизации путём рекомбинации, случайного изменения и отбора лучших решений-кандидатов. Данный подход был вдохновлён теорией эволюции и естественного отбора, изложенной в труде Чарльза Дарвина “Происхождение видов”.

Для решения задачи путём применения генетического алгоритма необходимо формализовать её таким образом, чтобы решение можно было представить в виде вектора “генов”. Обычно ген кодируется битом, но существуют вариации алгоритма, в которых он представляется в виде вещественного числа или некоторого другого объекта. Данные

вектора называют “особями” и представляют собой точку из пространства решений исследуемой задачи.

Далее необходимо ввести на множестве особей функцию-приспособленности. Эта функция является мерой “качества” решения задачи, закодированного данной особью.

На первом шаге генетического алгоритма инициализируется множество особей, как правило, случайным образом. После этого в цикле, до выполнения условия завершения алгоритма, начинают выполняться три генетических оператора: отбор, скрещивание и мутация. На рис. 1 представлена схема классического генетического алгоритма.

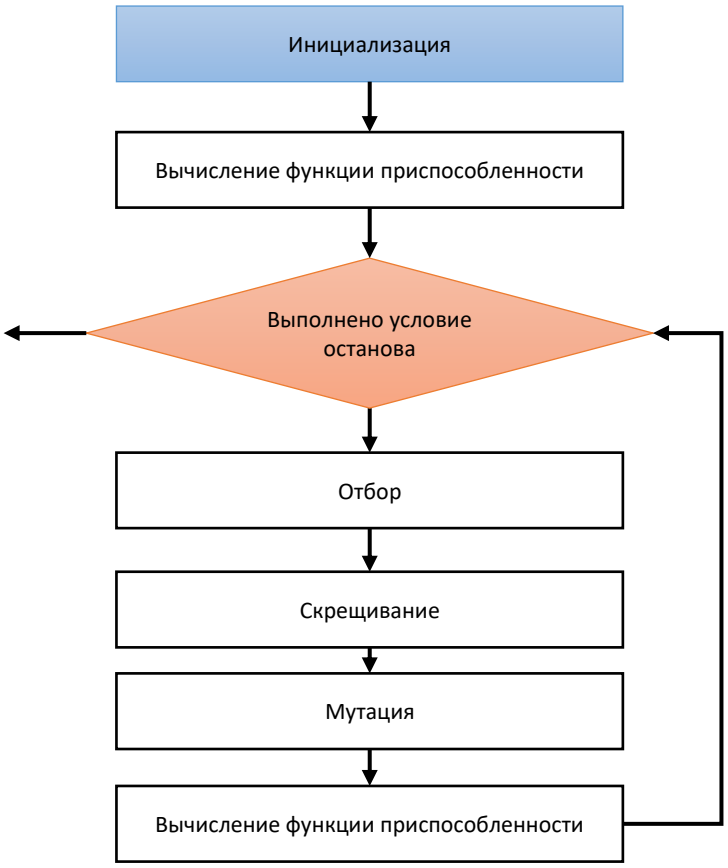


Рис. 1 Схема классического генетического алгоритма

На этапе отбора необходимо удалить из популяции определенную долю худших решений. Для каждой особи вычисляется значение функции приспособленности, после чего, в зависимости от вычисленных значений принимается решение о том, какие особи будут отсеяны. Существует несколько стратегий отбора, одна из них – турнирный отбор. Из популяции случайным образом выбираются группы из двух или более особей. В каждой группе отбор проходит только особь с наибольшим значением функции приспособленности.

Во время скрещивания из особей популяции на текущей итерации формируются новые особи-потомки, которые сочетают в себе информацию от своих предков. Определить данную операцию можно, например, следующим образом: из популяции выбираются две или более особи, их вектора генов делятся на несколько частей, после чего особь-потомок формируется объединением получившихся частей векторов генов различных предков. На рис. 2 показана схема операции скрещивания.

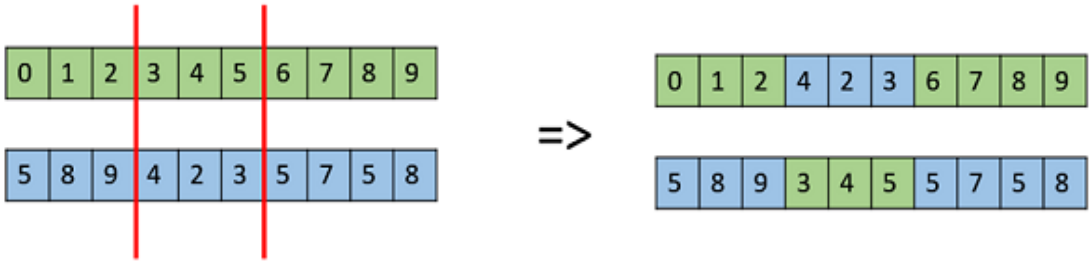


Рис. 2 Операция скрещивания

Операция мутации необходима для увеличения разнообразия популяции и исследования новых областей пространства поиска. Суть данной операции обычно заключается в случайном изменении векторов генов некоторых особей. Один из вариантов реализации – инверсия генов с некоторой заданной вероятностью.

В качестве условия остановки алгоритма могут выступать: достижение требуемой точности решения, ограничение на число итераций или ограничение на время работы алгоритма.

1.3 Понятие квантового кубита.

Обозначения Дирака (бра и кет) – это алгебраический формализм, предназначенный для описания векторов квантовых состояний.

$$|V\rangle = \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{pmatrix}$$

$$\langle W| = (w_1 \quad w_2 \quad \cdots \quad w_n)$$

Квантовый бит или кубит – это минимальный элемент для хранения информации в квантовом компьютере. Кубит также, как и классический бит, имеет два базовых состояния: $|0\rangle$ и $|1\rangle$, но в отличие от него может находиться в состоянии суперпозиции этих состояний. Суперпозиция базовых состояний записывается в виде их линейной комбинации: $|\Psi\rangle = \alpha|0\rangle + \beta|1\rangle$, где α и β – комплексные числа, называемые амплитудами вероятностей, отвечающие условию нормировки: $|\alpha|^2 + |\beta|^2 = 1$. При измерении кубит переходит из суперпозиции в одно из базовых состояний с вероятностями $|\alpha|^2$ и $|\beta|^2$ соответственно.

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

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

Изменение квантовой системы во времени описывается воздействием унитарного оператора U, называемого квантовым вентилем, на вектор состояния системы $|\Psi\rangle$.

$$|\Psi\rangle_t = U|\Psi\rangle_0$$

1.4 Модель генетического алгоритма, основанного на квантовом формализме.

Несмотря на название, генетический алгоритм, основанный на квантовом формализме, является классическим алгоритмом, предназначенным для исполнения на классическом компьютере. Использование терминов: «кубит», «амплитуда», «измерение» и других, обусловлено аналогией с соответствующими понятиями, используемыми в

квантовых вычислениях, и устоявшей терминологией в области эволюционных квантовых алгоритмов. Тем не менее, в дальнейшем, ради обеспечения краткости и читаемости, для обозначения данного алгоритма будет использовать название “квантовый генетический алгоритм” или сокращение КГА. Для классического генетического алгоритма может быть использовано сокращение ГА.

Основным отличием генетического алгоритма, основанного на квантовом формализме, от классического генетического алгоритма, является способ представления особей. В данном случае каждая особь кодируется не двоичным вектором, а вектором кубитов. Также вводится упрощение – в случае КГА кубит является парой вещественных чисел.

В силу вероятностной природы кубита каждая особь представляет собой не конкретное решение исследуемой задачи, а некоторое вероятностное распределение на множестве решений.

Эволюция особей происходит посредством применения к каждому кубиту их векторов унитарного оператора поворота в сторону лучшего найденного на данной итерации решения.

Наиболее популярная стратегия инициализации популяции – установка значения амплитуд всех кубитов особей так, чтобы вероятности их базовых состояний были равны. Такой способ инициализации отвечает равномерному вероятностному распределению на множестве решений. Также возможен вариант случайной инициализации.

На следующем шаге к популяции применяется операция измерения, в результате каждый кубит принимает базовое состояние 0 или 1 с вероятностями $|\alpha|^2$ и $|\beta|^2$ соответственно. Таким образом, каждой квантовой особи соответствует двоичный вектор наблюдения. На данных векторах вычисляются значения функции приспособленности и на их основании выбирается лучшая особь из текущей популяции. Если приспособленность данной особи больше, чем максимальное значение приспособленности, полученное за всё время работы алгоритма, то это значение и вектор наблюдения, на котором оно было получено, сохраняются.

После этого происходит проверка условия останова алгоритма. В качестве условий также могут выступать: число итераций алгоритма, время работы и заданная точность полученного решения.

Основным шагом квантового генетического алгоритма является применение оператора поворота к кубитам квантовых особей:

$$\begin{pmatrix} \alpha_{t+1}^{j} \\ \beta_{t+1}^{j} \end{pmatrix} = \begin{pmatrix} Cos(\theta_j) & -Sin(\theta_j) \\ Sin(\theta_j) & Cos(\theta_j) \end{pmatrix} \begin{pmatrix} \alpha_{t}^{j} \\ \beta_{t}^{j} \end{pmatrix}$$

$$\theta_j = s(\alpha_j, \beta_j)\delta$$

Функция $s(\alpha_j, \beta_j)$ отвечает за направление поворота и задается таким образом, чтобы увеличить амплитуду, соответствующую значению гена лучшего вектора-наблюдения.

В квантовом генетическом алгоритме также может применяться операция мутации. Реализовать её можно путем изменения порядка амплитуд кубита, изменения порядка кубитов особи или назначения кубиту некоторого случайного значения. Операция мутации применяется к кубитам с некоторой заданной вероятностью. В КГА, в отличие от ГА, применение мутации необязательно, так как в кубитах уже заложена случайность.

Существуют вариации квантового генетического алгоритма, называемые гибридным генетическим алгоритмом, в которых присутствует операция скрещивания. Она выполняется способом аналогичным тому, что был описан в классическом генетическом алгоритме.

Вероятностная природа кубитов позволяет считать, что каждая особь квантового генетического алгоритма отвечает сразу множеству решений. Данное свойства позволяет уменьшить число особей, используемых в алгоритме без потери качества решения и скорости сходимости. Следовательно, использование квантового генетического алгоритма потенциально может позволить уменьшить число вызовов функции приспособленности, наиболее вычислительно затратной части алгоритма, и объём памяти, требуемый для хранения особей, который может быть крайне велик при решении задач высоких размерностей.

1.5 Цель работы.

Целью данной выпускной квалификационной работы является исследование эффективности применения генетических алгоритмов, основанных на квантовом формализме, для настройки гиперпараметров алгоритмов машинного обучения на примере свёрточных нейросетевых моделей.

1.6 Постановка задачи.

- Провести обзор существующих подходов к применению квантового генетического алгоритма для решения задачи настройки гиперпараметров нейронных сетей.
- Разработать квантовый генетический алгоритм для настройки гиперпараметров свёрточных нейронных сетей.
- Исследовать эффективность разработанного квантового генетического алгоритма.

2. Обзор работ по применению квантового генетического алгоритма для решения задач оптимизации и подбора архитектур нейронных сетей.

В работе [1] квантовый генетический алгоритм применяется для решения задачи о 0-1 рюкзаке в следующей постановке.

Имеется m предметов, каждый из которых обладает некоторой ценностью p_i и весом w_i, где p_i и w_i – вещественные числа. Требуется подобрать подмножество предметов таким образов, чтобы максимизировать их общую ценность:

$$f(x) = \sum_{i=1}^{m} p_i x_i,$$

учитывая ограничение на суммарный вес предметов:

$$\sum_{i=1}^{m} w_i x_i \leq C,$$

где вместимость рюкзака C – положительное вещественной число.

Решения данной задачи кодируются в виде бинарного вектора, каждый бит которого отвечает некоторому предмету из заданного множества.

В процессе решения задачи о 0-1 рюкзаке генетическими алгоритмами могут быть сгенерированы особи, кодирующие решения, такие что суммарный вес предметов больше

чем вместимость рюкзака. Такие решения являются некорректными. Для обхода этой проблемы в классическом генетическом алгоритме могут быть применены различные механизмы:

- К функции приспособленности может быть добавлено слагаемое, отвечающее за штраф при получении некорректного решения. $f(x) = \sum_{i=1}^{m} p_i x_i - Pen(x)$. В описываемой статье использовалось 3 варианта штрафных слагаемых.
  - $Pen_1(x) = log_2(1 + \rho(\sum_{i=1}^{m} w_i x_i - C))$
  - $Pen_2(x) = \rho(\sum_{i=1}^{m} w_i x_i - C)$
  - $Pen_3(x) = \left(\rho(\sum_{i=1}^{m} w_i x_i - C)\right)^2$

  $\rho = max_{i=1\ldots m}\{\frac{p_i}{w_i}\}$
- Исправление особи. Если особь кодирует некорректное решения, функция приспособленности вычисляется для её исправленной версии. В данной статье предлагается 2 варианта метода исправления:
  - Rep_1. Последовательное удаление случайных предметов из рюкзака, пока он не перестанет быть переполненным.
  - Rep_2. Последовательное удаление предметов в порядке возрастания отношения их ценности к весу.
- Специальное проблемно-ориентированное кодирование особи.

В квантовом генетическом алгоритме применяется подход, основанный на исправлении особи. Сначала последовательно из рюкзака удаляются предметы, пока он не перестанет быть переполненным. Затем предметы последовательно добавляются, пока это возможно.

В вычислительном эксперименте были получены следующие результаты (табл. 1):

Кол-во предметов		ГА								КГА	
		Pen1	Pen2	Pen3	Rep1	Rep2	Dec1	Dec2	P2+R1	КГА(1)	КГА(10)
100	Лучшее	557.7	581.4	566.0	561.1	560.2	514.7	511.0	582.2	597.5	612.5
	Среднее	545.4	569.7	556.1	546.5	546.3	503.9	500.0	571.1	583.7	603.9
	Худшее	535.1	562.6	551.1	537.3	536.6	496.3	493.3	562.3	562.5	592.7
	Время	1.329	1.333	1.323	1.142	1.151	3.510	10.51	1.360	0.054	0.382
250	Лучшее	-	-	-	-	-	-	-	1391.9	1444.9	1480.3
	Среднее	-	-	-	-	-	-	-	1382.1	1412.4	1467.1
	Худшее	-	-	-	-	-	-	-	1364.8	1385.8	1443.8
	Время	-	-	-	-	-	-	-	3.292	0.141	1.380
500	Лучшее	-	-	-	-	-	-	-	2744.2	2824.1	2860.0
	Среднее	-	-	-	-	-	-	-	2720.8	2771.5	2841.3
	Худшее	-	-	-	-	-	-	-	2699.2	2744.3	2812.5
	Время	-	-	-	-	-	-	-	6.532	0.324	3.994

Табл. 1. ГА и КГА для решения задачи о 0-1 рюкзаке

Параметры эксперимента:

- Число запусков: 25.

- Количество поколений: 500.
- Объём популяции классического генетического алгоритма: 100.

Из полученных результатов видно, что при меньшем размере популяции, и как следствие. меньшем времени выполнения, результаты, достигнутые квантовым генетическим алгоритмом, превосходят результаты классического генетического алгоритма.

В статье [2] рассматривается применение квантового генетического алгоритма для решения задачи коммивояжёра. Постановка задачи.

Имеется полносвязный граф с m вершинами. Каждому ребру графа сопоставляется вес. Необходимо найти гамильтонов цикл этого графа такой, что суммарный вес всех, входящих в него рёбер, будет минимален.

При решении этой задачи с помощью квантового генетического алгоритма вводится особый вид кодировки особей, призванный учесть специфику задачи, связанную с тем, что ответ представляет собой упорядоченную последовательность вершин графа городов. Каждая особь кодируется матрицей Q размера $m \times m$:

$$Q = \begin{bmatrix} q_{11} & \cdots & q_{1m} \\ \vdots & \ddots & \vdots \\ q_{m1} & \cdots & q_{mm} \end{bmatrix}$$

элементы данной матрицы удовлетворяют условию нормировки по строкам:

$$\sum_{j=1}^{m} q_{ij} = 1, \forall\, i = \overline{1,n}$$

Значение элемента q_{ij} соответствует вероятности того, что город под номером j будет i-м посещённым городом.

Инициализация матрицы каждой особи происходит следующим образом:

$$Q(0) = \begin{bmatrix} 1 & 0 & & 0 \\ 0 & \frac{1}{n-1} & \cdots & \frac{1}{n-1} \\ & \vdots & \ddots & \vdots \\ 0 & \frac{1}{n-1} & \cdots & \frac{1}{n-1} \end{bmatrix}$$

При применении такой инициализации фиксируется начальный город, а остальные города получают одинаковые вероятности быть выбранными на том или ином шаге.

Предположим, что на втором шаге был выбран город с номером k_2. После этого, по условию задачи, город с этим номером не может быть выбран, поэтому необходимо обнулить соответствующий элемент в строках с номерами больше двух, а остальные элементы строк пропорционально изменить таким образом, чтобы они удовлетворяли условию нормировки:

$$\left(0 \quad \frac{q_{i2}}{1-q_{ik_2}} \quad \cdots \quad \frac{q_{i,k_2-1}}{1-q_{ik_2}} \quad 0 \quad \frac{q_{i,k_2+1}}{1-q_{ik_2}} \quad \cdots \quad \frac{q_{im}}{1-q_{ik_2}}\right), \forall\, i = \overline{2,m}$$

На дальнейших шагах построения последовательности городов выполняются аналогичные действия.

По полученной последовательности городов $(1 \quad k_2 \quad k_3 \quad \dots \quad k_{m-1} \quad k_m)$ можно построить матрицу $E(t)$ такую, что в $i-$ой строке на позиции k_i стоит 1, а на остальных нули.

Обновление весов происходит согласно уравнению:

$Q(t+1) = \big(1 - e(t)\big)Q(t) + e(t)E_{best}(t)$

$e(t)$ – скорость сходимости алгоритма.

$E_{best}(t)$ – матрица, построенная по лучшей последовательности, полученной за всё время работы алгоритма.

В статье представлены результаты эксперимента для 27, 48, 52 и 100 городов. Также были проведены сравнения результатов, количества вычислений целевых функций и времени исполнения упорядоченного квантового генетического алгоритма с аналогичной модификацией классического генетического алгоритма.

Применение квантового генетического алгоритма позволило достичь лучшего результата за счёт меньшего количества вычисления целевой функции, однако потребовало большего времени исполнения по сравнению в классическим генетическим алгоритмом. Вплоть до увеличения в 4 раза на конфигурации задачи с 100 городами.

В работе [3] рассмотрено применение квантового эволюционного алгоритма для решения задачи подбора архитектуры сверточной нейронной сети. Предложенная авторами идея состоит в следующем.

Кодировка квантовой особи состоит из двух частей. Первая отвечает за кодировку архитектурных параметров: типы слоёв и их параметры. Вторая отвечает за кодирование численных параметров обучения, таких как параметр скорости обучения алгоритма оптимизации весов нейронной сети.

Гены части квантовой особи, отвечающей за кодирование архитектурных параметров, представляют собой набор m вещественных чисел $(q_1 \quad q_2 \quad \cdots \quad q_m)$, удовлетворяющих правилу нормировки $\sum_{i=1}^{m} q_i = 1$. Результатом измерения такого гена с вероятностью q_i будет число i для $i = \overline{1,m}$. Эволюция гена происходит за счёт увеличения вероятности q_j при условии, что значение соответствующего гена у вектора-наблюдения, имеющего наибольшее значение функции приспособленности, равно j. Для

удовлетворения условия нормировки необходимо пропорционально уменьшить остальные амплитуды гена.

Гены части квантовой особи, отвечающие за кодирование численных параметров обучения, описываются в виде пары вещественных чисел, являющихся нижней и верхней границей равномерного распределения. Результатом измерения такого гена будет вещественное число из равномерного распределения, закодированного в данном гене. Эволюция гена происходит за счёт смещения математического ожидания распределения, закодированного в гене, к значению соответствующего гена у вектора-наблюдения, имеющего наибольшее значение функции приспособленности, и уменьшения дисперсии этого распределения.

Вектора-измерения получаются путём конкатенации результатов измерений двух частей квантовой особи.

При проведении вычислительного эксперимента для каждой квантовой особи генерировалось 4 вектора-наблюдения. После чего к векторам-наблюдениям применялась операция скрещивания по случайной двоичной маске. Гены архитектурной части квантовой особи обновляются раз в несколько поколений с вероятностью, являющейся параметром алгоритма. Количество поколений также является параметром.

Гены архитектурной части квантовых особей кодировали один из приведённых в таблице 2 слоёв.

Слой	Размер ядра	Сдвиг	Кол-во выходных фильтров	Начальная вероятность
ConvBlock	1	1	32	0.042
ConvBlock	1	1	64	0.042
ConvBlock	3	1	32	0.042
ConvBlock	3	1	64	0.042
ConvBlock	3	1	128	0.042
ConvBlock	3	1	256	0.042
ConvBlock	5	1	32	0.042
ConvBlock	5	1	64	0.042
MaxPool	2	2	-	0.167
AvgPool	2	2	-	0.167
NoOp	-	-	-	0.333

Табл. 2. Слои, кодируемые архитектурными генами квантовой особи

Также в таблице указаны начальные значения амплитуд. Тип слоя ConvBlock означает последовательную цепочку слоёв, состоящую из свёрточного слоя, слоя батч-нормализации и функции активации ReLU. Тип слоя NO_OP необходим для возможности кодирования архитектур переменной длины.

Описанные в статье эксперименты были отнесены к двум типам. В экспериментах первого типа настраивались только архитектурные параметры, во втором - и архитектурные и численные параметры обучения. В качестве алгоритма оптимизации весов нейронной сети рассматривался алгоритм RMSProp. В таблице 3 отображены конфигурации параметров вычислительных экспериментов.

Параметр	Эксперимент 1	Эксперимент 2
decay	0.9	[0.1, 0.999]
momentum	0.0	[0.0, 0.999]
weight decay	1.0e-4	[1.0e-5, 1.0e-3]
crossover rate	-	0.5
generations	300	
network nodes	20	
quantum individuals	5	
repetition	4	
update generations	5	
update quantum rate	0.1	

Табл. 3. Параметры экспериментов

Нейронные сети обучались на усечённом наборе данных CIFAR-10. 9000 изображений использовались для обучения и 1000 изображений для контроля. Для окончательной оценки лучшая найденная архитектура обучалась на полном наборе данных. В таблице 4 предоставлены результаты вычислительного эксперимента.

№	Эксперимент	Время	Число слоёв	Точность		
				Усечённый CIFAR-10	Полный CIFAR-10 (обучение)	Полный CIFAR-10 (тест)
1	1	56 ч.	13	0.7840	0.8852	0.8773
2	1	52 ч.	11	0.7740	0.8748	0.8812
3	1	61 ч.	12	0.7790	0.9008	**0.8928**
4	1	54 ч.	10	0.7910	0.8944	0.4925

5	1	56 ч.	15	0.7820	0.8958	0.8874
1	2	52 ч.	12	0.7730	0.8852	0.8765
2	2	52 ч.	11	0.7750	0.8648	0.8620
3	2	54 ч.	16	0.7770	0.8796	0.8805
4	2	51 ч.	15	0.7670	0.8728	0.8575
5	2	51 ч.	11	0.7850	0.8408	0.8430

Табл. 4 Результаты эксперимента

В результате проведения вычислительного эксперимента удалось достичь точность классификации в 89% за 51 GPU/день. Этот результат сравнивают с результатом из статьи [4], где подбор архитектур для задачи CIFAR-10 осуществлялся с помощью классических эволюционных алгоритмов. Им удалось достичь 94% точности за 2700 GPU/дней. Отношение разницы времени выполнения к разности в точности позволяет говорить о потенциале решения задачи подбора архитектур нейронных сетей с помощью квантовых эволюционных алгоритмов.

В работе [5] также рассматривается применение квантового генетического алгоритма для решения задачи подбора архитектур сверточных нейронных сетей.

Гены квантовых особей кодируются двумя амплитудами, следовательно, вектора-наблюдения являются двоичными векторами. Они в свою очередь кодируют последовательность слоёв и их параметры. В таблицах 5-7 приведено описание слоёв и их конфигураций, примеры кодировок слоёв, параметры вычислительного эксперимента.

Тип слоя	Параметр	Число бит	Диапазон значений
Свёрточный	Размер ядра	3	[1, 8]
	Число выходных фильтров	7	[1, 128]
	Сдвиг	2	[1,4]
Pooling – слой	Размер ядра	2	[1, 4]
	Сдвиг	2	[1, 4]
	Тип: 1 (maximal), 2 (average)	1	[1, 2]
	Пропуск	6	-
Полносвязный	Число выходных нейронов	11	[1, 2048]

Пропуск слоя	Пропуск	11	-

Табл. 5. Кодируемые слои и их конфигурации

Тип слоя	Кодировка
Свёрточный	(0000) 001 0001111 01
Pooling	(00010) 01 01 0 001111
Полносвязный	(00011) 01111111111
Пропуск слоя	(00100) 01111111111

Табл. 6. Примеры кодировок слоев

Параметр	Значение
Количество запусков	10
Размер популяции	30
Число поколений	30
Число эпох обучения нейронных сетей	10
Максимальное число свёрточных слоёв в сети	10
Максимальное число полносвязных слоёв в сети	3
Размер батча обучения	128
Функция активации в сетях	ReLU
Оптимизатор весов	Adam

Табл. 7. Параметры вычислительного эксперимента

В процессе вычислительного эксперимента нейронные сети обучались на трех наборах данных: MNIST, MNISTRB и CS. Полученные результаты сравнивались с результатами, которых удалось достичь используя популярные архитектуры свёрточных нейронных сетей и архитектуру, сгенерированную другим алгоритмом подбора архитектур нейронных сетей IPPSO, основанным на алгоритме роя частиц. Полученные результаты приведены в таблицах 8 и 9.

Нейронная сеть	MNIST	MNISTRB	CS
CAE-2	97.52	54.77	-
RandNet-2	98.75	56.31	95.55

PCANet-2	98.60	54.14	95.81
LDANet-2	98.95	61.46	92.78
DBN-3	96.54	52.61	81.37
SAA-3	96.54	48.07	81.59
NNet	95.31	37.84	67.75
SVM-RBF	96.97	44.82	80.87
SVM-Poly	96.31	43.59	80.18
КГА (среднее)	98.82	79.80	93.31
КГА (стандартное отклонение)	0.097	1.54	1.63

Табл. 8. Точность ручных архитектур и КГА

Набор данных	Лучший		Среднее		Стандартное отклонение	
	IPPSO	КГА	IPPSO	КГА	IPPSO	КГА
MNIST	98.95	98.97	98.79	98.82	0.103	0.097
MNISTRB	67.50	80.19	65.20	79.80	2.96	1.54
CS	91.52	94.52	87.94	93.31	2.25	1.63

Табл. 9. Точность IPPSO и КГА

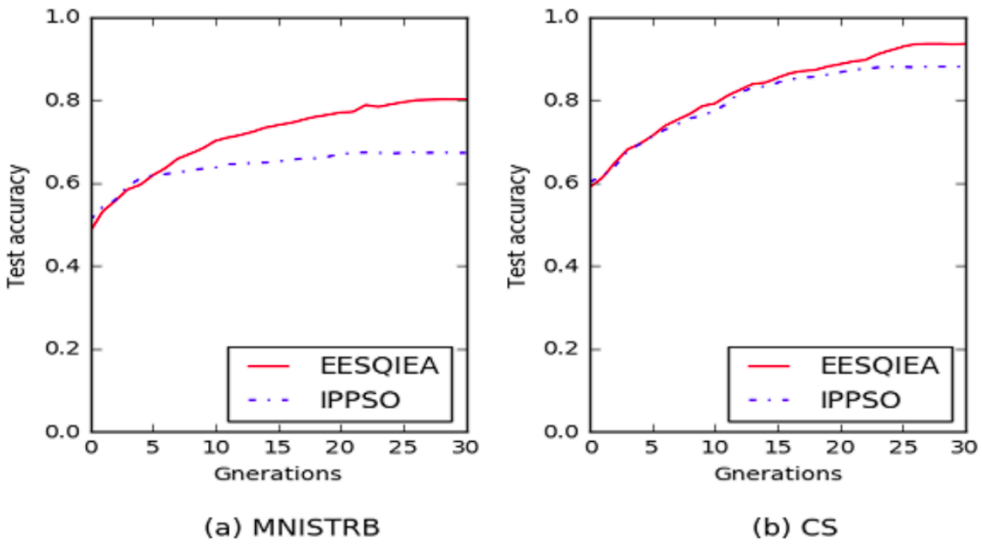


Рис. 3. Точность IPPSO и КГА

На рисунке 3 показаны графики зависимости точности от итерации КГА и IPPSO га наборах данных MNISTRB и CS.

По результатам вычислительного эксперимента видно, что квантовый генетический алгоритм способен генерировать решения, которые по качеству превосходят некоторые

решения, спроектированные вручную, или сгенерированные эволюционным алгоритмом IPPSO.

3. Разработка квантового генетического алгоритма для настройки гиперпараметров сверточных нейронных сетей.

На основе проведённого анализа существующих решений задачи настройки параметров нейросетей в работе предлагается метод построения квантового генетического алгоритма для конструирования архитектуры и настройки гиперпараметров свёрточных нейросетей. Для исследования эффективности предложенного метода предлагается провести сравнительный анализ нейросетей, построенных с использованием разработанного квантового генетического алгоритма, с сетями, построенными на основе классического генетического алгоритма. Сравнительный анализ сетей предлагается провести на примере решения задачи классификации изображений.

Как квантовый, так и классический генетические алгоритмы обладают высоким потенциалом для эффективной параллельной реализации. Это связано с тем, что некоторые генетические операторы, такие как мутация, применение оператора поворота к кубитам квантовой особи или измерение квантовой особи выполняются независимо для каждой особи. Стоит отдельно заметить, что вычисление функции приспособленности (качества) особей популяции также происходит независимо и является наиболее вычислительно ёмкой частью работы алгоритма во многих задачах. В работе рассматриваются подходы к распараллеливанию предложенных КГА и ГА. Предлагаемый в работе метод распараллеливания основан на параллельном вычислении функции качества сразу для нескольких особей.

При решении задачи оптимизации гиперпараметров нейросетевых моделей в качестве значения функции приспособленности выступает некоторая мера качества нейронной сети, обученной с использованием данной конфигурации гиперпараметров. Как известно, процесс обучения нейронных сетей требует значительных вычислительных затрат. Для сокращения времени обучения предлагается рассмотреть возможность использования графических ускорителей. Таким образом, параллельные вычисления функции качества дополняются переносом части вычислений на GPU.

Для того, чтобы обусловить применение квантового и классического генетических алгоритмов для решения задачи подбора архитектур нейронных сетей, предлагается также провести сравнительный анализ разработанных алгоритмов с методом случайного поиска.

3.1 Библиотека функций для построения генетических алгоритмов.

Разработка генетических алгоритмов требует определения метода формального описания особей популяции, выбора функции качества, описывающей приспособленность каждой особи к решению поставленной задачи, методов реализации эволюционных операторов отбора, скрещивания, и мутации.

В данном разделе рассматриваются предлагаемые решения для построения квантового генетического алгоритма. Для исследования эффективности классический и квантовый генетические алгоритмы были реализованы в виде библиотек libClassicGen и libQuantGen на языке C++.

3.1.1 Классический генетический алгоритм:

- Особи кодируются целочисленными векторами типа **char**.
- Функция приспособленности может настраиваться пользователем и является параметром алгоритма.
- Оператор отбора происходит по средствам разбития всей популяции на пары. Вероятность получить ту или иную конфигурацию пар не зависит от значений функций приспособленности особей, то есть всевозможные конфигурации пар равновероятны.
- Оператор мутации осуществляется путем инверсии генов с некоторой вероятностью. Вероятность задается через параметр алгоритма.
- Оператор скрещивания выполняется между парой особей. Двойки для скрещивания отбираются за счет разбития всей популяции на пары. Процесс разбития популяции происходит аналогично тому, что был описан в операторе отбора. Участок скрещивания представляет из себя двоичную маску. Она формируется в два этапа. На первом этапе из случайного равномерного распределения генерируется число $r \in [0; 0{,}5]$. На втором этапе генерируется двоичная маска, длина которой совпадает с длиной особи. Каждый бит маски принимает значение 0 и 1 с вероятностями $1 - r$ и r соответственно.

Скрещивание в каждой паре происходит с некоторой вероятностью. Вероятность задается через параметр алгоритма. В листинге 1 показан алгоритм генерации случайной маски участка скрещивания, используемый в реализованных ГА и КГА.

```
char* mask = new char[individ_size];
std::mt19937_64 rand;
std::uniform_real_distribution<> dist(0, 1);
double prob = dists(rand) / 2;
for (int j = 0; j < individ_size; j++) {
    if (dists(rand) < prob) {
        mask[j] = 1;
    } else {
        mask[j] = 0;
    }
}
```

Листинг 1. Генерация двоичной маски участка скрещивания

- Условием останова алгоритма является заданное число поколений.

3.1.2 Квантовый генетический алгоритм:

- Особи кодируются векторами объектов класса **QuantBit**. Класс **QuantBit** предназначен для кодирования генов квантовых особей. Содержит два числа типа **double** для хранения амплитуд, операцию инверсии для выполнения мутации и операцию поворота на заданный угол для обновления квантовых особей.
- Функция приспособленности и функция вычисления угла поворота могут настраиваться пользователем и являются параметрами алгоритма.
- Оператор мутации выполняется аналогично тому, как он выполнялся в классическом генетическом алгоритме. Есть возможность исключить его из алгоритма, задав соответствующий параметр.
- Оператор скрещивания выполняется аналогично тому, как он выполнялся в классическом генетическом алгоритме. Есть возможность исключить его из алгоритма, задав соответствующий параметр.
- Условием останова алгоритма является заданное число поколений.

На рисунке 1 показана схема предложенного квантового генетического алгоритма.

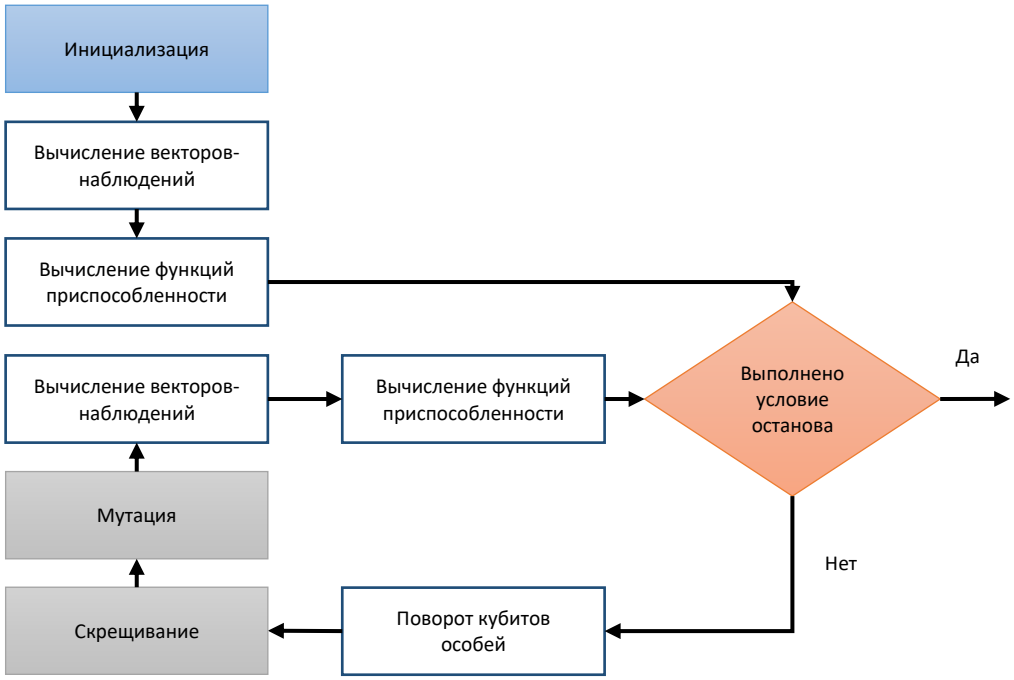


Рис. 4. Схема предлагаемого КГА

3.2 Параллельные реализации классического и квантового генетических алгоритмов.

Основными технологиями для параллельных реализаций алгоритмов были выбраны MPI и OpenMP.

Особи популяции распределяются равномерно между доступными узлами. Операции мутации, вычисления векторов-наблюдений, поворота кубитов квантовых особей и вычисления функций приспособленности могут быть выполнены независимо для каждой особи, поэтому их исполнение переносится в параллельные секции OpenMP. Для выполнения операции отбора в классическом генетическом алгоритме и операции скрещивания в обоих вариантах генетического алгоритма возможны ситуации, при которых происходит взаимодействие особей, находящихся на разных узлах. В этот случае требуются обмены данными между узлами.

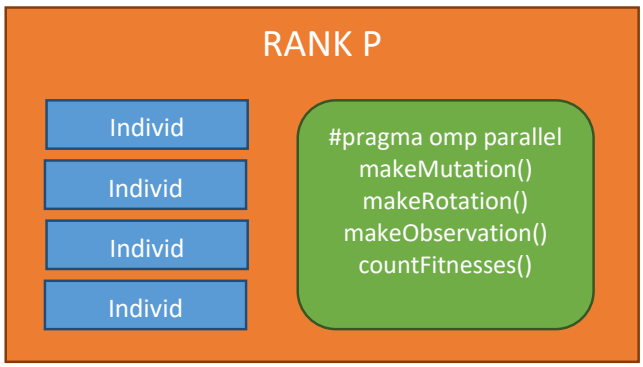


Рис. 5. Расположение особей

3.2.1 Параллельный классический генетический алгоритм. Предлагаемая реализация операции отбора.

На первом шаге выполнения операции отбора классического генетического алгоритма значения функций приспособленности всех особей популяции пересылаются на процесс с рангом 0 с помощью функции MPI_Gatherv.

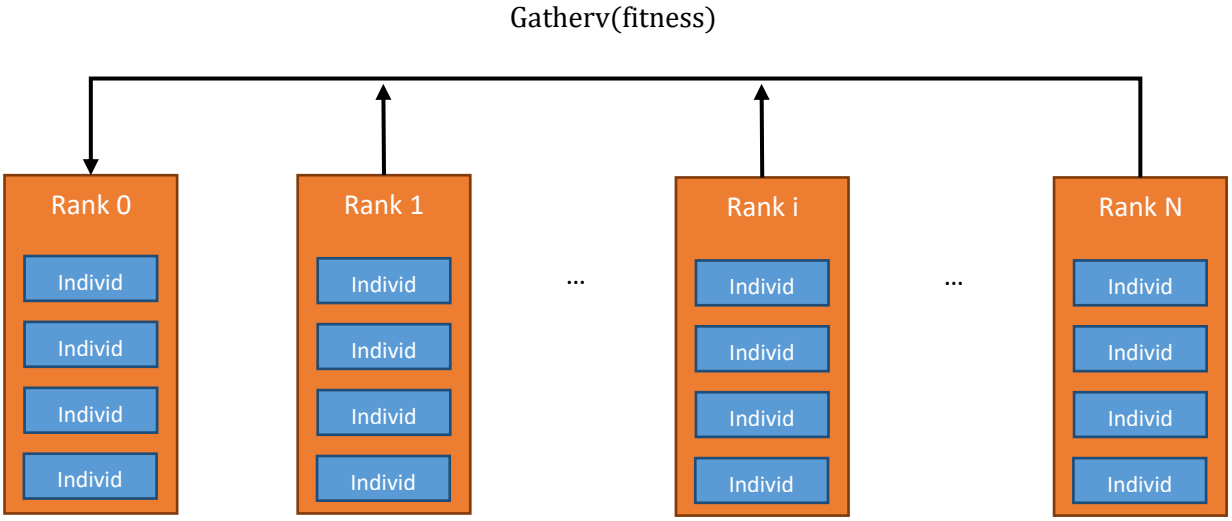


Рис. 6. Схема операции отбора. Этап 1

На втором этапе нулевой процесс разбивает популяцию на пары случайным образом, в каждой паре на первое место ставится индекс особи, имеющей большее значение функции приспособленности среди особей данной пары. Далее каждая пара отсылается на процессы, содержащие особи, входящие в данную пару.

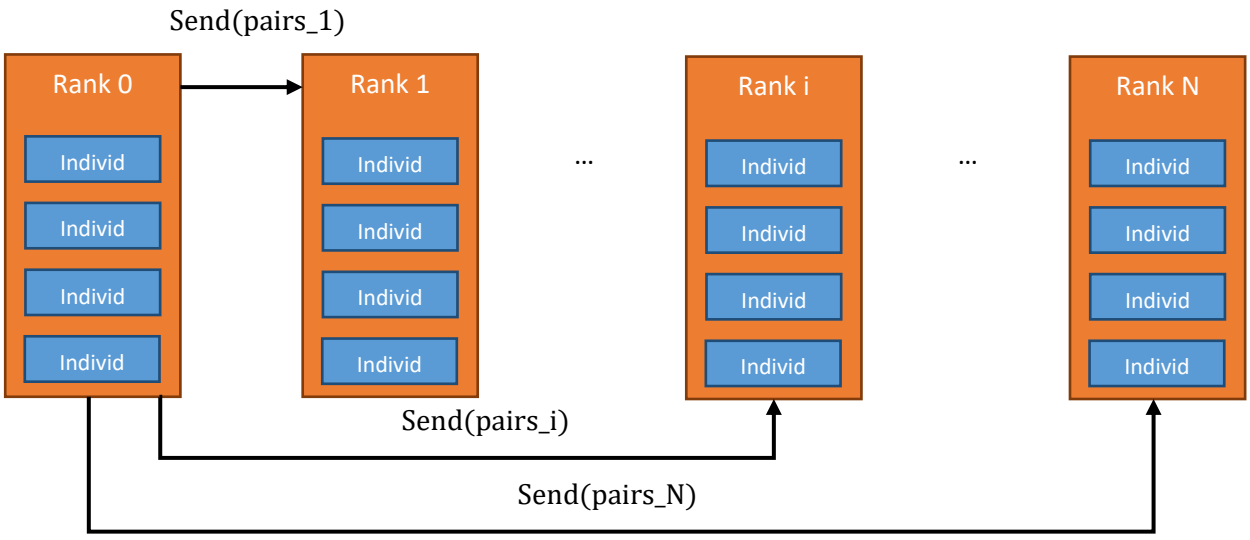


Рис. 7. Схема операции отбора. Этап 2

На третьем шаге каждый процесс для всех полученных пар:

- Если обе особи, входящие в пару, лежат на данном процессе, гены первой особи пары копируются во вторую особь. Пересылок между процессами не производится.
- Если процессу принадлежит особь, находящаяся в паре на первом месте, эта особь пересылается процессу, которому принадлежит вторая особь.
- Если процессу принадлежит особь, находящаяся в паре на втором месте, то процесс принимается первую особь и копирует её гены во вторую.

Чтобы в процессе пересылки особей пары не перемешивались, при вызовах функций MPI_Send и MPI_Recv параметр tag устанавливается равным индексу отсылаемой/принимаемой особи.

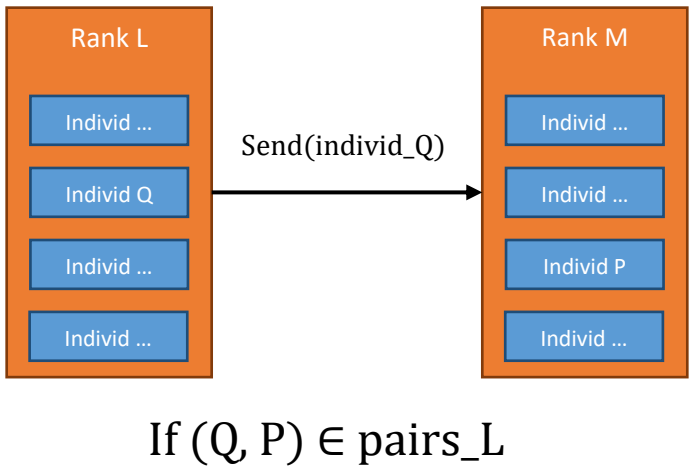


Рис. 8. Схема операции отбора. Этап 3

3.2.2 Параллельный классический генетический алгоритм. Предлагаемая реализация операции скрещивания.

На первом этапе выполнения операции скрещивания нулевой процесс случайном образом разбивает популяцию на пары. Затем он рассылает сформированные пары на остальные процессы аналогично тому, как это делалось при выполнении операции отбора.

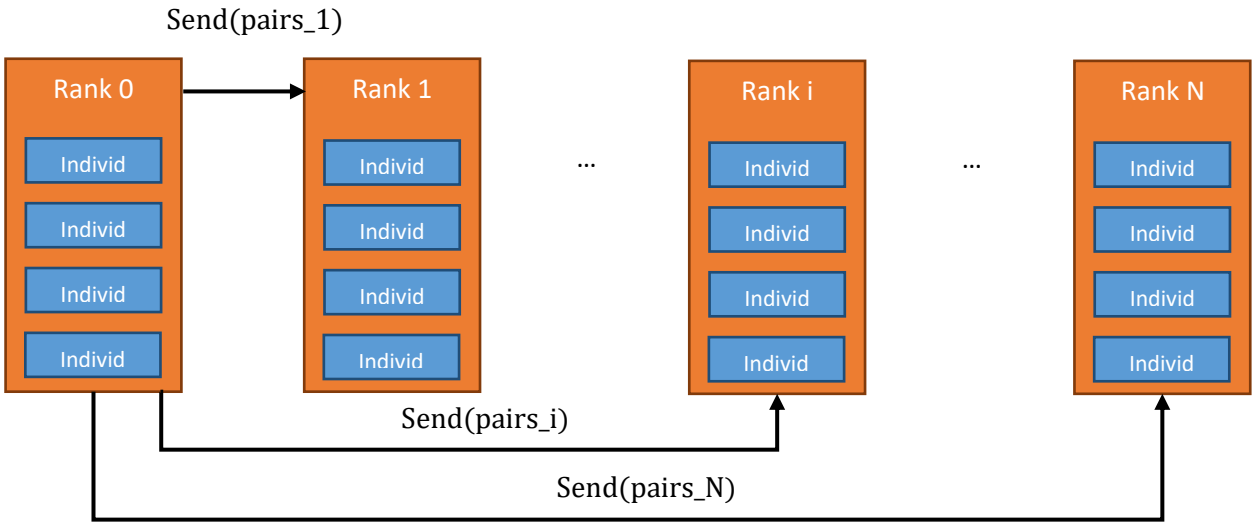
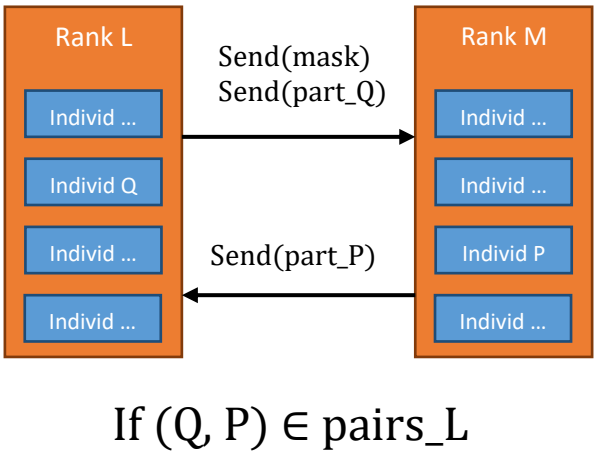


Рис. 9. Схема операции скрещивания. Этап 1

На втором шаге процесс, содержаний особь пары с меньшим индексом, формирует случайную двоичную маску, обозначающую участок скрещивания. Если особи пары находятся на разных процессах, то процесс, сформировавший маску, отсылает ее и участок скрещивания своей особи на процесс, содержащий вторую особи. Далее он дожидается участка скрещивания второй особи. В противном случае обмен участками скрещивания между особями происходит локально.



If (Q, P) ∈ pairs_L

Рис. 10. Схема операции скрещивания. Этап 2

3.2.3 Параллельный квантовый генетический алгоритм. Предлагаемая реализация операции обновления лучшего вектора-наблюдения.

С помощью функции MPI_Allreduce(…, MPI_MAXLOC, …) все процессы получаются информацию о значении функции приспособленности лучшего вектора-наблюдения на текущей итерации и его расположении. Затем, если полученное значение функция приспособленности больше максимального за все время работы алгоритма, происходит обновление максимального достигнутого значения функции приспособленности на всех процессах и коллективная пересылка нового лучшего вектора-наблюдения с помощью функции MPI_Bcast.

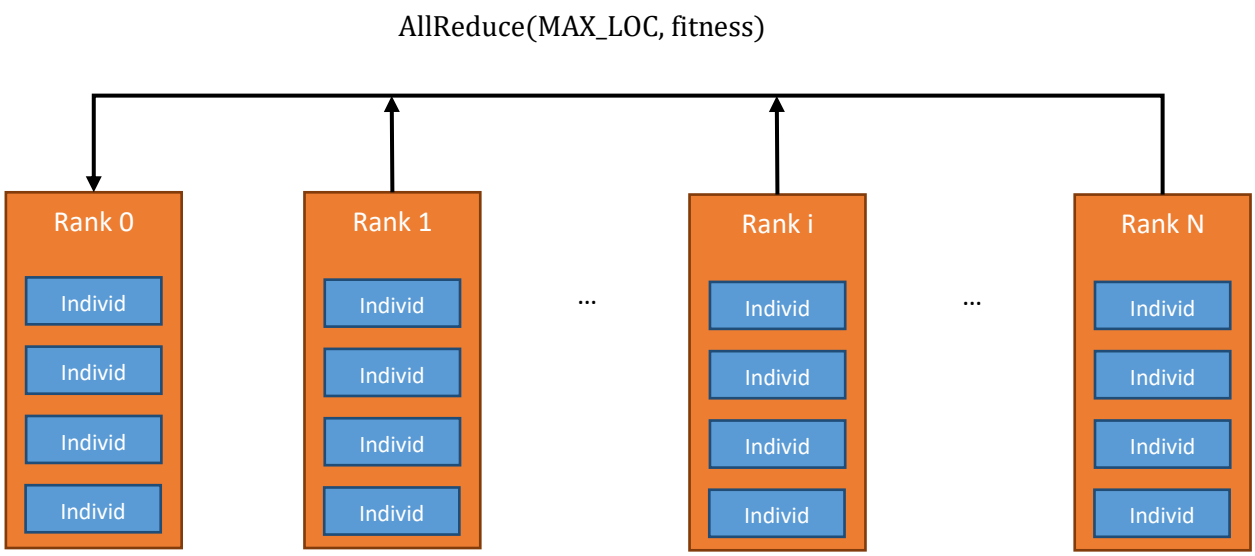


Рис. 11. Схема операции обновления лучшего вектора-наблюдения. Этап 1

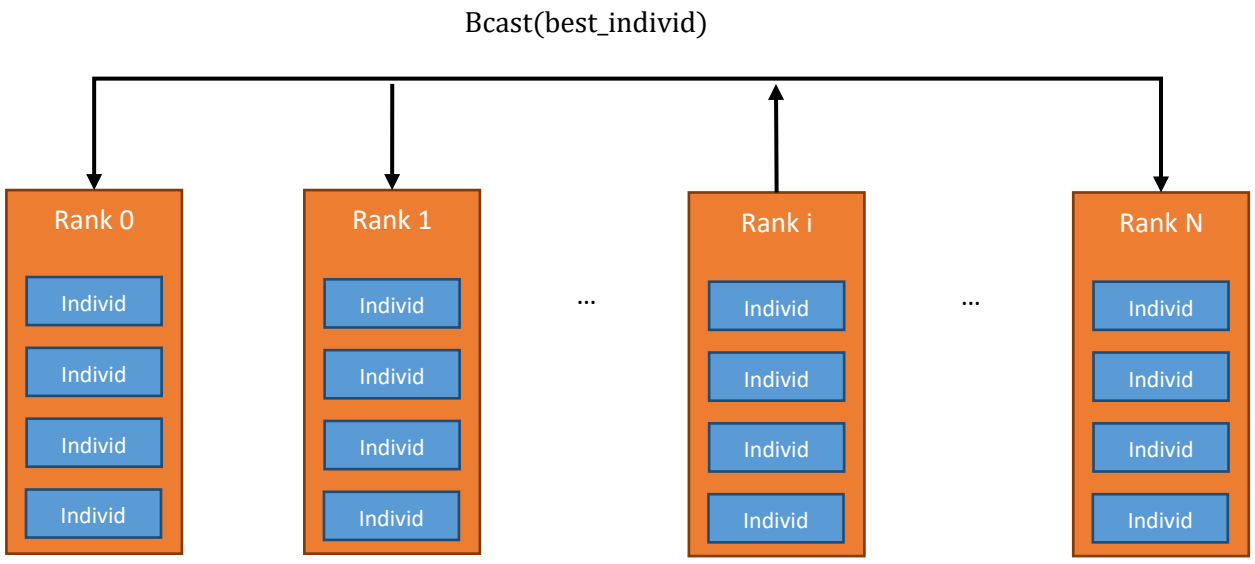


Рис. 12. Схема операции обновления лучшего вектора-наблюдения. Этап 2

Параметр	Тип	Описание
global_pop_size	int	Размер популяции
individ_size	int	Длина индивида
need_mutation (только КГА)	bool	Наличие операции мутации в КГА
mutation_probability	double	Вероятность мутации
need_crossover (только КГА)	bool	Наличие операции скрещивания в КГА
crossover_probability	double	Вероятность скрещивания
fitness_function	double(char*, int)	Функция приспособленности
angle_function (только КГА)	double(char, char, double, double, QuantBit)	Функция угла поворота в КГА
mpi_size	int	Число MPI процессов
omp_size	int	Число потоков OMP
seed	int	Инициализация генератора случайных чисел

Табл. 10. Параметры реализованных ГА и КГА

3.3 Метод настройки гиперпараметров сверточной нейронной сети с использованием ГА и КГА.

В процессе решения задачи настройки гиперпараметров нейронных сетей с помощью классического и квантового генетических алгоритмов подбирались:

- Архитектурные параметры:
  - Число слоев
  - Типы слоев
  - Параметры слоев
- Параметры обучения:
  - Алгоритм оптимизации весов нейронной сети.
  - Параметр алгоритма оптимизации – скорость обучения.

Архитектура нейронной сети кодируется двоичным вектором. В начале данного вектора находится блок бинарных строк, кодирующих архитектуру. Каждая из этих строк

отвечает за отдельный слой нейронной сети. Слои в архитектуре соединяются последовательно в том же порядке, в котором идут соответствующие им строки в итоговой кодировке.

Длина строки, кодирующей слой, - 14. Первые 2 бита определяют тип слоя, остальные 12 – параметры специфичные для каждого слоя. Исключением является первый слой, кодировка которого состоит из 12 бит. Это связано с тем, что было принято решение рассматривать только архитектуры, включающие свёрточные слои. Таким образом, кодируемая архитектура всегда начинается со свёрточного слоя.

В нейронной сети могут присутствовать слои четырех типов: свёрточный, pooling-слой, полносвязный и пропуск слоя. Тип слоя «пропуск» является служебным и необходим для возможности кодирования архитектур переменной длины.

На кодируемые архитектуры нейронных сетей вводится техническое ограничение: после первого полносвязного слоя могут идти только полносвязные слои. Это ограничение выполняется за счет того, что после первого появления в кодировке полносвязного слоя, код, соответствующий свёрточному слою и pooling-слою, начинает восприниматься как полносвязный слой.

В конец архитектуры добавляется полносвязный слой, количество выходных нейронов которого совпадает с количеством классов решаемой задачи классификации.

Последние 9 бит кодировки отвечают за конфигурацию алгоритма оптимизации весов нейронной сети. 2 бита отводится на тип оптимизатора, 7 на его параметр – скорость обучения.

При кодировании использовались коды Грэя.

В таблице 11 приведена подробная конфигурация настраиваемых параметров.

Группа параметров	Параметр	Число бит	Диапазон значений
Свёрточный слой	Размер ядра (NxN)	3	[1, 8]
	Величина сдвига	2	[1, 4]
	Число выходных фильтров	5	[1, 32]
	Функция активации	2	{Sigmoid, ReLU, Tanh, SoftPlus}
Pooling слой	Размер ядра (NxN)	2	[1, 4]

	Сдвиг	2	[1, 4]
	Пропуск	8	-
Полносвязный слой	Число выходных нейронов	10	[1, 1024]
	Функция активации	2	{Sigmoid, ReLU, Tanh, SoftPlus}
Пропуск слоя	Пропуск	12	-
Алгоритм оптимизации весов	Тип оптимизатора	2	{SGD, Adagrad, RMSprop, Adam}
	Скорость обучения	7	[1 * 1e-4, 128 * 1e-4]

Табл. 11. Спецификация настраиваемых гиперпараметров

3.4 Программная реализация процесса конструирования и обучения нейронных сетей.

Обучение нейронных сетей проводилось с помощью библиотеки PyTorch для языка C++: libtorch версии v1.3.1.

Программный файл train.cpp содержит функции для декодирования кодов Грэя, функции для конструирования слоёв сети на основе кодировки, класс Net, наследованный от внутреннего класс библиотеки libtorch torch::nn::Module и функцию train, содержащую логику процесса обучения.

В листинге 2 приведен код функции преобразования кодов Грэя в десятичное число.

```
int bin2dec(const char* bin, int size) {
    int res = 0;
    int pow = 1;
    for (int i = size - 1; i >= 0; i--) {
        res += pow * bin[i];
        pow *= 2;
    }

    return res;
}
```

```
int gray2dec(const char* gray, int size) {
    char bin[size];
    char value = gray[0];
    bin[0] = value;
    for (int i = 1; i < size; i++) {
        if (gray[i] == 1) {
            value = (value + 1) % 2;
        }
        bin[i] = value;
    }
    return bin2dec(bin, size);
}
```

Листинг 2. Функция преобразования кода Грэя в десятичное число

Функции конструирования слоёв принимают на вход код слоя и возвращают отвечающий ему объект библиотеки libtorch. Также в процессе работы этих функций изменяется состояние объекта класс Info. Объект класса Info содержится в классе Net и хранит некоторую служебную информацию, необходимую для корректной интерпретации кодировки.

В листинге 3 приведен код функции, отвечающей за декодирование полносвязного слоя.

```
torch::nn::Linear construct_fc(const char* code, Info& info) {
    const int size_size = 10;
    const int act_func_stride = size_size;
    const int act_func_size = 2;

    int size = decoding_function(
        code,
        size_size
    ) + 1;
    int act_func = decoding_function(
        code + act_func_stride,
        act_func_size
    );

    torch::nn::Linear res = nullptr;
    if (info.only_fc) {
```

```
        res = torch::nn::Linear(info.out_size, size);
    } else {
        res = torch::nn::Linear(
            info.pic_channels * info.pic_size * info.pic_size,
            size
        );
    }
    info.fc_count++;

    info.out_size = size;
    info.only_fc = true;
    info.layers[info.layers_count++] = 2;

    info.act_funcs[info.act_funcs_count++] = act_func;

    return res;
}
```

Листинг 3. Функция конструирования полносвязного слоя

Класс Net содержит информацию об архитектуре сети и значениях её весов. Он наследуется от класса torch::nn::Module для того, чтобы использовать возможности автоматического дифференцирования и оптимизации весов нейронной сети, которые предоставляет библиотека libtorch.

Функция train содержит в себе код, описывающий создание сети и процесс обучения. Возвращает точность классификации обученной сети на тестовой выборке. Для решения задачи подбора архитектуры нейронной сети с помощью разработанных ГА и КГА должна быть передана в генетические алгоритмы в качестве параметра (функция приспособленности). Для согласования списка параметров функции train и параметра ГА и КГА fitness_function необходимо связать некоторые параметры функции train, используя std::bind.

Библиотека libtorch предоставляет средства для переноса данных и проведения расчётов на GPU. Таким образом, процесс обучения происходит с использованием графических ускорителей.

4. Вычислительный эксперимент.

4.1 Набор данных.

В процессе вычислительного эксперимента нейронные сети обучались на наборе данных CIFAR-10. Он состоит из 60000 трехканальных изображений размера 32x32 пикселя, каждое из которых отнесено к одному из 10 классов. На рисунке 13 приведены примеры изображений из набора данных CIFAR-10.

50000 изображений использовались для обучения, 10000 для контроля.

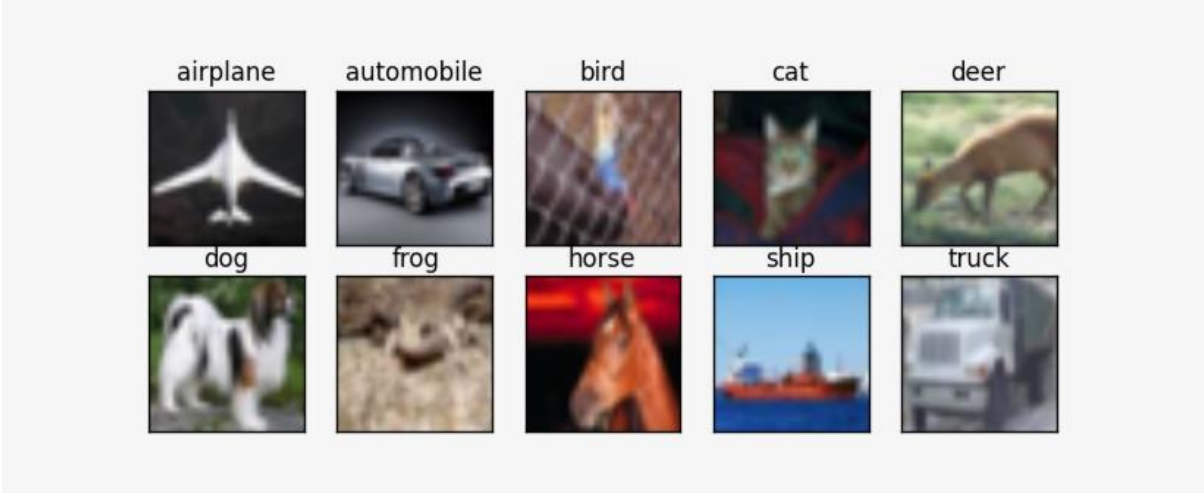


Рис. 11. примеры изображений из набора данных CIFAR-10

В качестве функции приспособленности была выбрана доля правильно классифицированных изображений из тестовой выборки.

4.2 Алгоритмы и их конфигурации.

Для решения задачи подбора архитектур нейронных сетей были применены: классический генетический алгоритм, квантовый генетический алгоритм и метод случайного поиска.

Объем популяции классического и квантового генетических алгоритмов был равен 16. Классический генетический алгоритм запускался с четырьмя конфигурациями параметров: вероятность мутации, вероятность скрещивания.

Вероятность мутации	Вероятность скрещивания
0.05	0.5
0.1	0.5
0.05	0.75
0.1	0.75

Рис. 12. Конфигурации параметров ГА

Квантовый генетический алгоритм запускался без использования операторов мутации и скрещивания.

4.3 Аппаратная и программная платформы, использованные в вычислительном эксперименте.

Эксперимент проводился на вычислительной системе Polus. Для вычислений были задействованы два узла с выделением двух ядер на каждом из них. Каждый из четырех получившихся потоков использовал отдельный графический ускоритель.

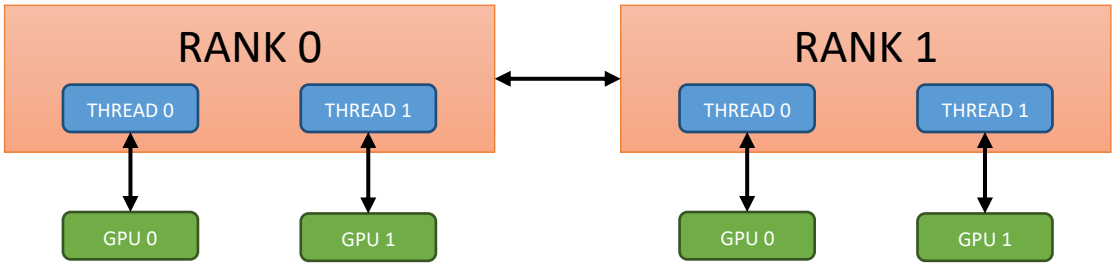


Рис. 134. Конфигурация аппаратной платформы вычислительного эксперимента

Время исполнения каждого из алгоритмов ≈ 3 часа.

5. Результаты вычислительного эксперимента.

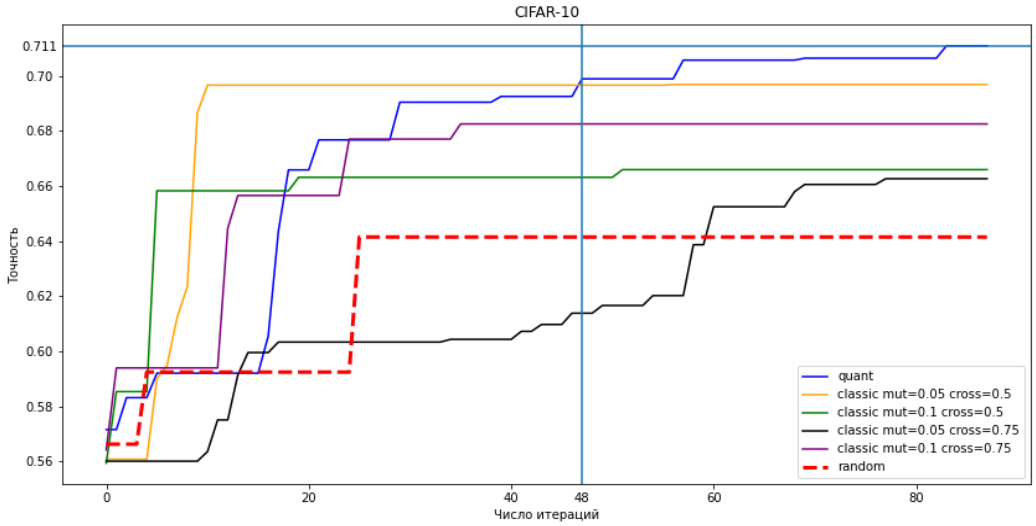


Рис. 15. Зависимость максимальной достигнутой точности нейронных сетей на наборе данных CIFAR-10 от итерации ГА и КГА

На рис. 15 приведены графики значений максимальной достигнутой точности классического генетического алгоритма с четырьмя конфигурациями параметров, квантового генетического алгоритма и метода случайного поиска.

Как можно видеть результаты ГА и КГА превосходят результаты метода случайного поиска и одинаковом количестве вычислений функции приспособленности. Необходимо отметить, что в данном эксперимент количество вычислений функции приспособленности является показательной метрикой эффективности алгоритма, так как эта операция во много раз вычислительно сложнее остальных операций рассматриваемых алгоритмов.

В свою очередь квантовый генетический алгоритм показывает более высокие результаты, чем классический генетический алгоритм, начиная с итерации номер 48. При этом можно заметить, что даже на последних итерация КГА происходит уточнение лучшего решения, в то время, как лучший из ГА сошелся к локальному максимуму уже в районе 10 итерации.

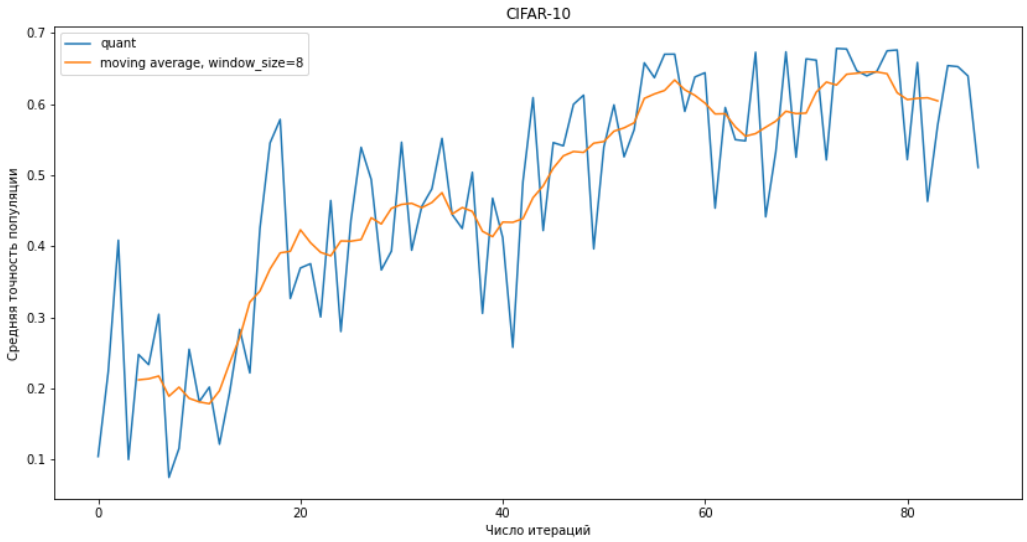


Рис. 16. Зависимость средней точности нейронных сетей от итерации КГА

На рис. 16 изображена зависимость средней по популяции точности нейросетей от итерации для КГА. Как можно видеть из данного графика, наблюдается не только увеличение максимальной достигнутой точности, но и улучшение качества особей популяции в среднем.

На рис. 17 показано время работы квантового генетического алгоритма при решении задачи подбора архитектуры нейронной сети на наборе данных CIFAR-10.

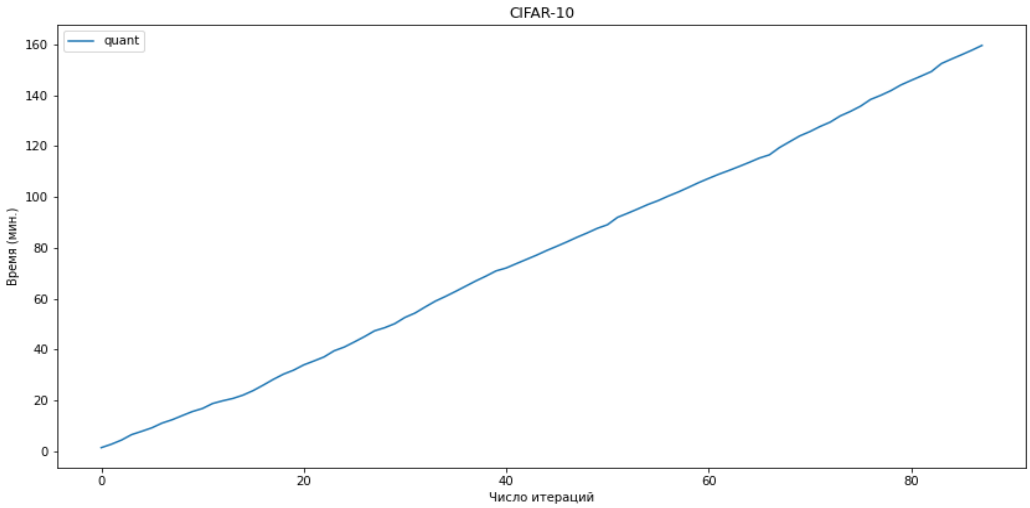


Рис. 1714. Время исполнения КГА при решении задачи подбора архитектуры нейронной сети на наборе данных CIFAR-10

6. Заключение.

Основные результаты, полученные в работе, заключаются в следующем.

- Проведен обзор существующих подходов к применению квантового генетического алгоритма для решения задач оптимизации и, в частности, для решения задач подбора гиперпараметров нейросетевых моделей.
- Предложены параллельные классический генетический алгоритм и квантовый генетический алгоритм. На основе разработанных алгоритмов создана библиотека, включающая в свой состав параллельные реализации ГА и КГА на основе технологий MPI и OpenMP.
- Разработан метод настройки гиперпараметров свёрточной нейронной сети. Реализация метода выполнена с использованием разработанной библиотеки.
- Проведен эксперимент с использованием параллельных вычислительных ресурсов системы Polus.
- Проведён сравнительный анализ результатов, полученных путём применения ГА, КГА и метода случайного поиска к настройке параметров свёрточной нейронной сети для анализа изображений на примере набора данных CIFAR-10. Показано преимущество КГА по сравнению с другими рассмотренными методами в применении к рассмотренной задаче.

7. Возможные пути дальнейших исследований:

- Исследовать влияние операций скрещивания и мутации на эффективность КГА.
- Исследовать эффективность применения квантовых эволюционных алгоритмов с вещественной кодировкой (Real-Coded Quantum-Inspired Evolutionary Algorithm) для решения задачи подбора гиперпараметров алгоритмов машинного обучения.

8. Список литературы.

[1] Kuk-Hyun Han, Jong-Hwan Kim. Genetic Quantum Algorithm and its Application to Combinatorial Optimization Problem [Электронный ресурс]. – Электрон. дан – URL: https://www.researchgate.net/publication/3865219_Genetic_quantum_algorithm_and_its_application_to_combinatorial_optimization_problem. (дата обращения: 14.05.2021).

[2] Luciano R. Silveira, Ricardo Tanscheit, Marley Vellsaco. Quantum-Inspired Genetic Algorithms applied to Ordering Combinatorial Optimization Problems [Электронный ресурс]. – Электрон. дан – URL: https://www.researchgate.net/publication/261304673_Quantum-inspired_genetic_algorithms_applied_to_ordering_combinatorial_optimization_problems. (дата обращения: 14.05.2021).

[3] Daniela Szwarcman, Daniel Civitarese, Marley Vellasco. Quantum-Inspired Neural Architecture Search [Электронный ресурс]. – Электрон. дан – URL: https://www.researchgate.net/publication/336156892_Quantum-Inspired_Neural_Architecture_Search. (дата обращения: 14.05.2021).

[4] Esteban Real, Sherry Moore, Andrew Selle, Saurabh Saxena, Yutaka Leon Suematsu, Jie Tan, Quoc V. Le, Alexey Kurakin. Large-Scale Evolution of Image Classifiers [Электронный ресурс]. – Электрон. дан – URL: https://arxiv.org/abs/1703.01041. (дата обращения: 14.05.2021).

[5] Weiliang Ye, Ruijiao Liu, Yangyang Li, Licheng Jiao. Quantum-Inspired Evolutionary Algorithm for Convolutional Neural Networks Architecture Search [Электронный ресурс]. – Электрон. дан – URL: https://ieeexplore.ieee.org/abstract/document/9185727. (дата обращения: 14.05.2021).