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

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

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

Выполнил:
студент 423 группы
Васильев Семён

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

Оптимизация гиперпараметров моделей машинного обучения

Гиперпараметры в машинном обучении - это параметры, значения которых используется для управления процессом обучения.

Значения устанавливаются перед запуском процесса обучения. В этом смысле они и отличаются от обычных параметров, вычисляемых в процессе обучения.

Популярные подходы к решению задачи подбора гиперпараметров:

- Поиск по решётке.
- Случайный поиск.
- Байесовская оптимизация.
- Эволюционные алгоритмы.

Актуальность

- Активное использование алгоритмов машинного обучения при решении современных научных и прикладных задач.
- Существенное влияние гиперпараметров на качество алгоритма машинного обучения.
- Высокая вычислительная сложность процесса обучения моделей машинного обучения.
- Необходимость использования параллельных вычислительных систем для сокращения времени, необходимого для подбора оптимальной конфигурации гиперпараметров.

Классический генетический алгоритм

Имеется **популяция особей**. Каждая особь – двоичный вектор, кодирующий решение задачи.

На множестве особей определена вещественная **функция приспособленности**, определяющая меру качества решения.

К особям итеративно применяются эволюционные операторы:

- **Отбор**. Замена менее приспособленных особей более приспособленными.
- **Мутация**. Случайное изменение генов особей.
- **Скрещивание**. Обмен информацией между особями.

Генетический алгоритм, основанный на квантовом формализме. Кодирование особей.

КГА отличается от ГА способом кодирования особи и набором генетических операторов.

- Имеется популяция особей.
- Каждая особь – вектор генов.
- Ген - пара вещественных чисел α и β, удовлетворяющих условию нормировки: $|\alpha|^2 + |\beta|^2 = 1$.
- Операция «измерения» гена возвращает 0 или 1 с вероятностями $|\alpha|^2$ и $|\beta|^2$ соответственно.

$$\begin{pmatrix} \alpha_1 & \alpha_2 & \alpha_3 & \dots & \alpha_j \\ \beta_1 & \beta_2 & \beta_3 & \dots & \beta_j \end{pmatrix}_1 \begin{pmatrix} \alpha_1 & \alpha_2 & \alpha_3 & \dots & \alpha_j \\ \beta_1 & \beta_2 & \beta_3 & \dots & \beta_j \end{pmatrix}_2 , \dots, \begin{pmatrix} \alpha_1 & \alpha_2 & \alpha_3 & \dots & \alpha_j \\ \beta_1 & \beta_2 & \beta_3 & \dots & \beta_j \end{pmatrix}_i$$

Для краткости, будет использоваться название “**квантовый генетический алгоритм**” или сокращение **КГА**.
Для генетического алгоритма (классического) будет использоваться сокращение **ГА**.

Операции, применяемые в КГА

- Инициализации популяции: $\alpha_i = \frac{1}{\sqrt{2}},\ \beta_i = \frac{1}{\sqrt{2}}$
- Ко всем генам особи применяется операция «измерения». Результат – двоичный вектор-наблюдение.
- Функция приспособленности – функция, заданная на векторах-наблюдениях.
- К генам особей популяции применяется оператор поворота в сторону вектора-наблюдения с наибольшим значением функции приспособленности.

Достоинства КГА

- Применение КГА для решения классических задач оптимизации (задача коммивояжера, задача о рюкзаке и др.) позволило получить результаты лучше, чем применение ГА.
- Из-за вероятностной природы генов квантовой особи каждая особь КГА представляет сразу множество решений.
- Хороший потенциал для параллельной реализации.

Цель

- **Цель работы** — исследование эффективности применения генетических алгоритмов, основанных на квантовом формализме, для настройки гиперпараметров алгоритмов машинного обучения на примере свёрточных нейросетевых моделей.

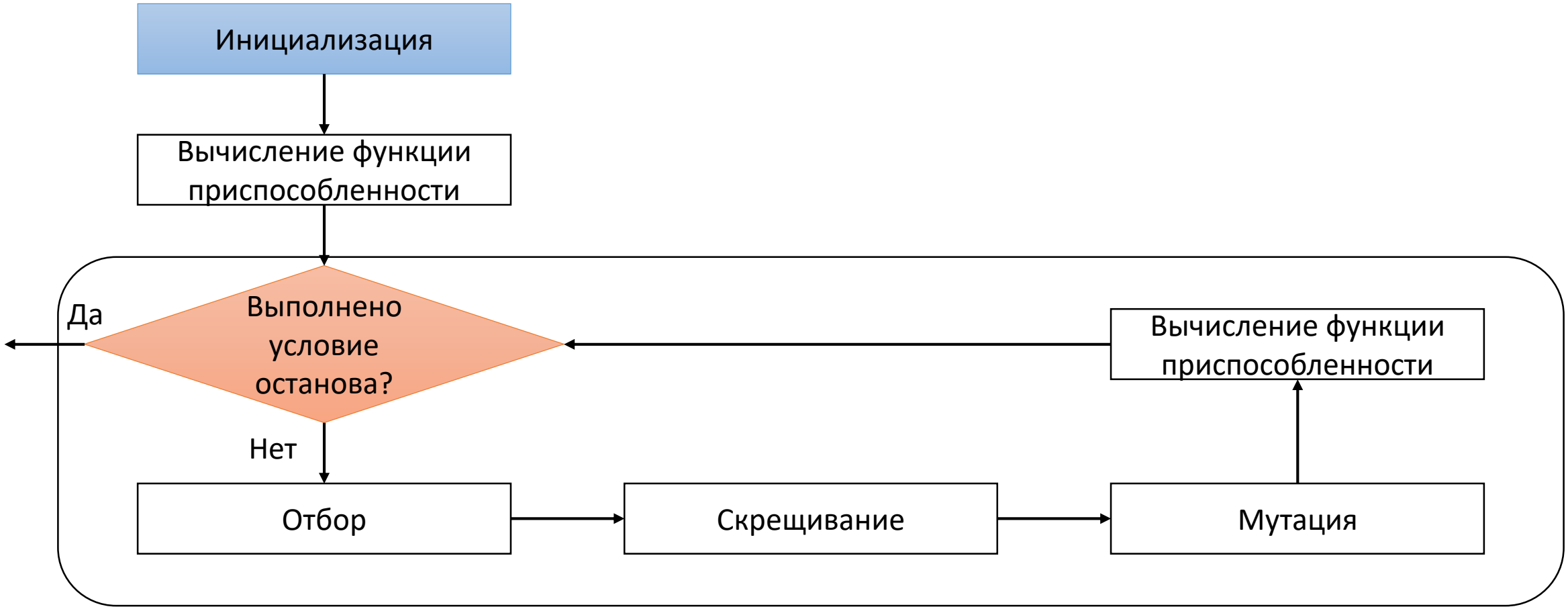
Постановка задачи

- Провести обзор существующих подходов к применению КГА для решения задачи настройки гиперпараметров нейросетей.
- Разработать КГА для настройки гиперпараметров свёрточных нейронных сетей.
- Исследовать эффективность разработанного КГА.

Предлагаемый подход к решению задачи

- Разработка ГА, КГА и метода случайного поиска для решения задачи оптимизации гиперпараметров нейросетей.
- Разработка параллельной библиотеки для построения нейронных сетей с использованием предложенных алгоритмов.
- Проведение сравнительного анализа разработанных алгоритмов для оптимизации гиперпараметров свёрточных нейронных сетей для классификации изображений.

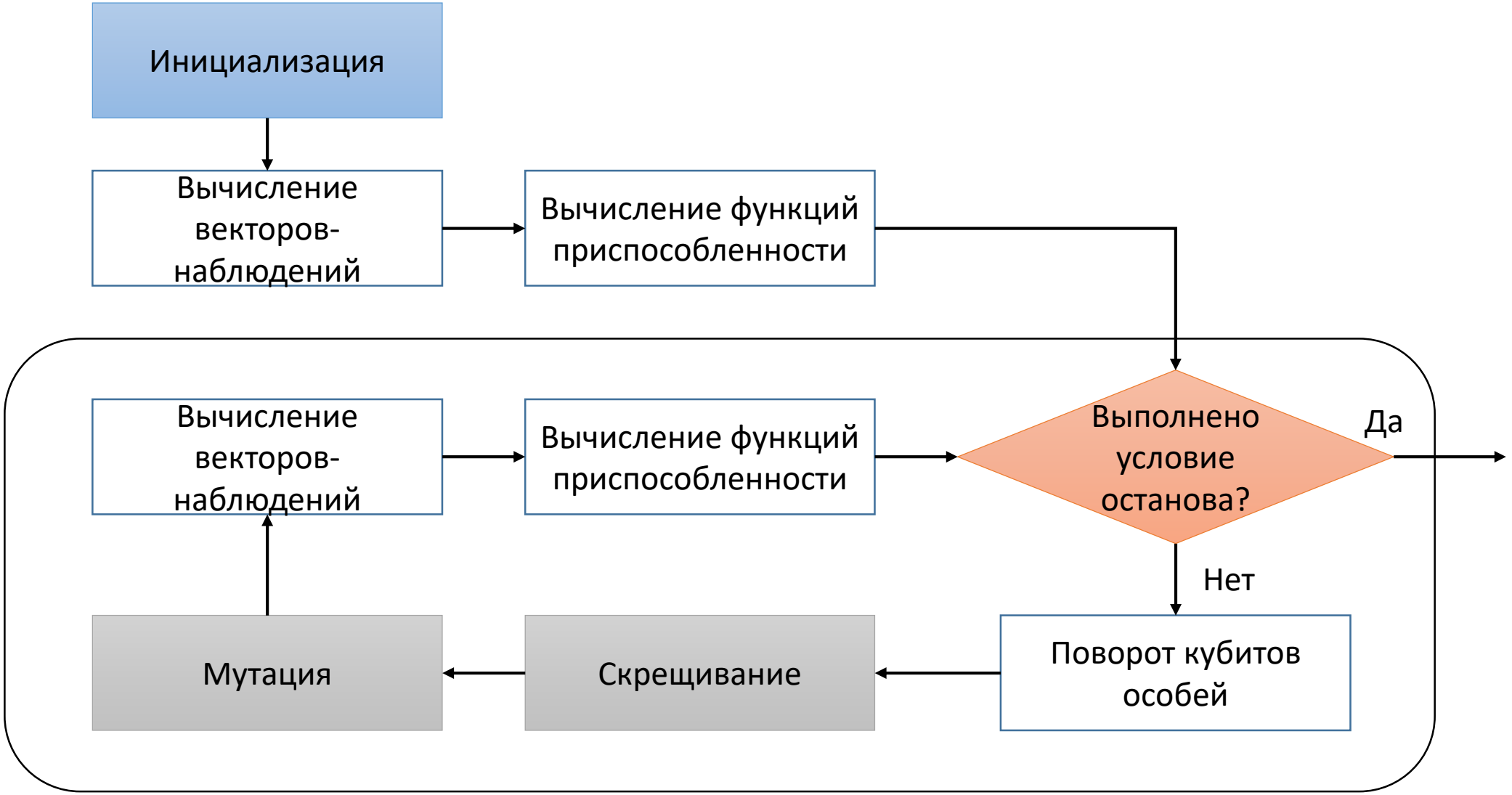
Схема предложенного ГА



Реализация предложенного ГА

- **Кодирование особей**: вектора типа **char**.
- **Функция приспособленности**: передаётся через параметр:
  - std::function<double(const char*, int)> fitness_function.
- **Оператор отбора**: разбиение популяции на пары.
- **Оператор мутации**: инверсия генов с заданной через параметр вероятностью.
- **Оператор скрещивания**:
  - Участок для скрещивания – случайная двоичная маска.
  - Разбиение популяцию на пары.
  - Скрещивание в каждой паре выполняется с заданной вероятностью.
- **Условие останова:** заданное число итераций.

Схема квантового генетического алгоритма



Реализация предложенного КГА

- **Кодирование особей:** вектор пар вещественных чисел.
- **Функция приспособленности и функция поворот**: передаются через параметры:
  - std::function<double(char*, int)> fitness_function
  - std::function<double(char, char, double, double, QuantBit)> angle_function
- **Оператор мутации**: инверсия генов с заданной через параметр вероятностью. Может быть исключена пользователем из алгоритма.
- **Оператор скрещивания**:
  - Участок для скрещивания – случайная двоичная маска.
  - Разбиение популяцию на пары.
  - Скрещивание в каждой паре выполняется с заданной вероятностью.
  - Может быть исключена пользователем из алгоритма.
- **Условие останова:** заданное число итераций.

Интерфейс реализованных библиотек. Конструкторы

QuantGenAlg::QuantGenAlg(…) / ClassicGenAlg::ClassicGenAlg(…)

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

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.

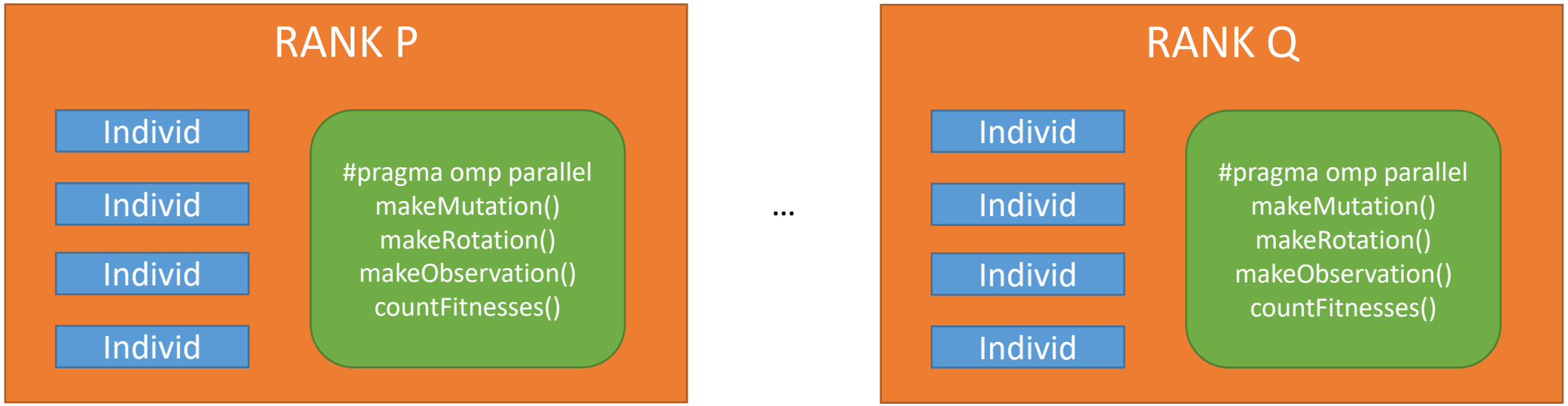
Интерфейс реализованных библиотек. ГА

- ClassicGenAlg
  - **void** countFitnesses() – вычисление функций приспособленности всех особей.
  - **void** makeSelection() – выполнение операции отбора.
  - **void** makeMutation() – выполнение операции мутации всех особей.
  - **void** makeCrossover() – выполнение операции скрещивания.
  - **void** startAlgorithm(**int** iter_num) – выполнение алгоритма по предложенной схеме.
  - **void** printPopulation() – вывод популяции.
  - **void** printParams() – вывод параметров алгоритма.

Интерфейс реализованных библиотек. КГА

- QuantGenAlg
  - **void** makeObservations() – вычисление векторов-наблюдений всех особей и функций приспособленности на них.
  - **void** makeRotation() – применение операторов поворота ко всех генам всех особей.
  - **void** makeMutation() – выполнение операции мутации всех особей.
  - **void** makeCrossover() – выполнение операции скрещивания.
  - **void** startAlgorithm(**int** iter_num) – выполнение алгоритма по предложенной схеме.
  - **void** printPopulation() – вывод популяции.
  - **void** printParams() – вывод параметров алгоритма.

Параллельная реализация ГА и КГА



Операции отбора и скрещивания в ГА и операция обновления лучшего вектора-наблюдения в КГА требуют обмены данными.

Алгоритмы реализованы на языке C++ с использованием технологий MPI/OpenMP.

Метод случайного поиска

- На каждой итерации генерируется множество случайных двоичных векторов из равномерного распределения на пространстве поиска.
- Такой алгоритм эквивалентен ГА с параметрами:
  - Вероятность мутации: 0.5
  - Вероятность скрещивания: 0
- Этот подход выбран для того, чтобы эксперименты с использование генетических алгоритмов и методом случайного поиска имели одинаковую структуру.

Настройка гиперпараметров свёрточной нейронной сети. Список параметров.

- **Архитектурные**:
  - Число слоёв.
  - Типы слоёв.
  - Параметры слоёв
- **Параметры обучения**:
  - Тип оптимизатора.
  - Скорость обучения.

Спецификация настраиваемых гиперпараметров.

Группа параметров	Параметр	Число бит	Диапазон значений
Сверточный слой	Размер ядра (NxN)	3	[1, 8]
	Величина сдвига	2	[1, 4]
	Число фильтров на выходе	5	[1, 32]
	Функция активации	2	{Sigmoid, ReLU, Tanh, SoftPlus}
Pooling слой	Размер ядра (NxN)	2	[1, 4]
	Величина сдвига	2	[1, 4]
	Пропуск	8	-
Полносвязный слой	Число нейронов на выходе	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]

Кодировка параметров.

- Размер кодировки одного слоя – 14 битов.
- Первые 2 бита – тип слоя. Остальные 12 – параметры слоя.
- Генетические алгоритмы работают с особями фиксированной длины. По этой причине перед запуском алгоритма необходимо определить глубину подбираемых нейронных сетей. За счет специального типа слоя «пропуск» появляется возможность кодировать сети переменной длины (меньшей или равной определенной при запуске).
- Первый слой – свёрточный. Последний – полносвязный. После первого полносвязного только полносвязные.

Последние 9 битов отвечают за конфигурацию алгоритма оптимизации весов нейронной сети.

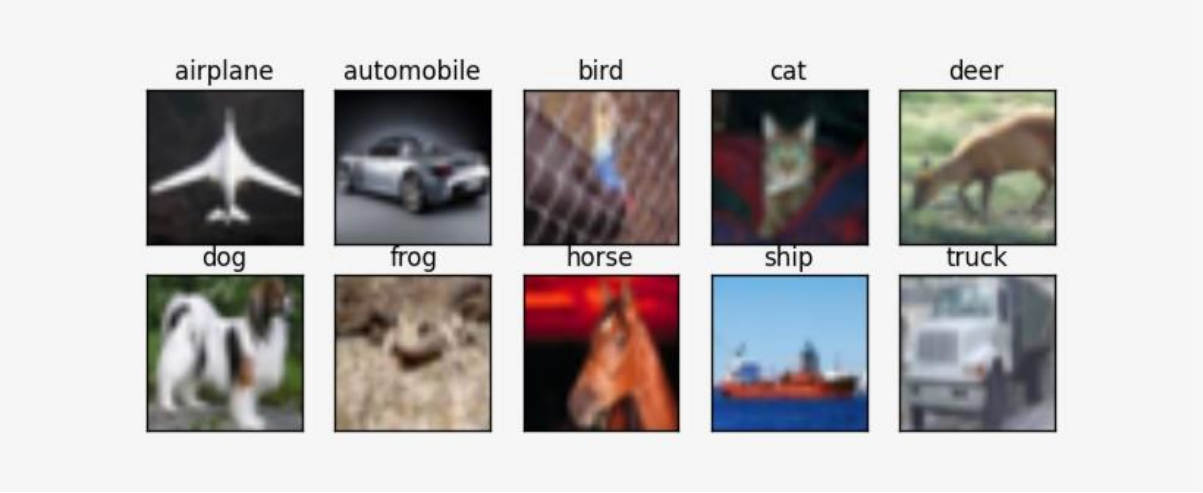
При кодировании использовался код Грэя.

Вычислительный эксперимент.

Набор данных, используемый при проведении вычислительного эксперимента

Нейронные сети обучались на наборе данных CIFAR-10:

- 60000 изображений, 3 канала, размер 32x32.
- 10 классов.
- 50000 изображений для обучения, 10000 для контроля.



Функции приспособленности - доля правильно классифицированных изображений (точность) из тестовой выборки.

Вычислительный эксперимент.

Значения параметров ГА и КГА и метода случайного поиска.

Размер популяции: 16 особей.

Конфигурации параметров ГА:

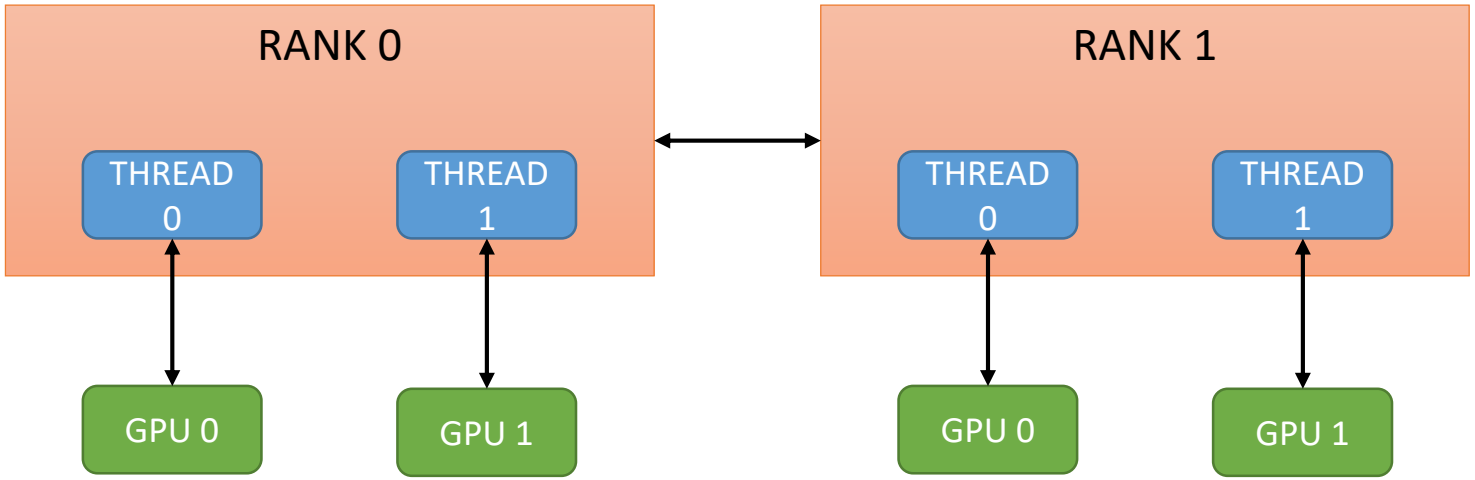
Вероятность мутации	Вероятность скрещивания
0.05	0.5
0.1	0.5
0.05	0.75
0.1	0.75

Конфигурации параметров ГА.

КГА без мутации и скрещивания.

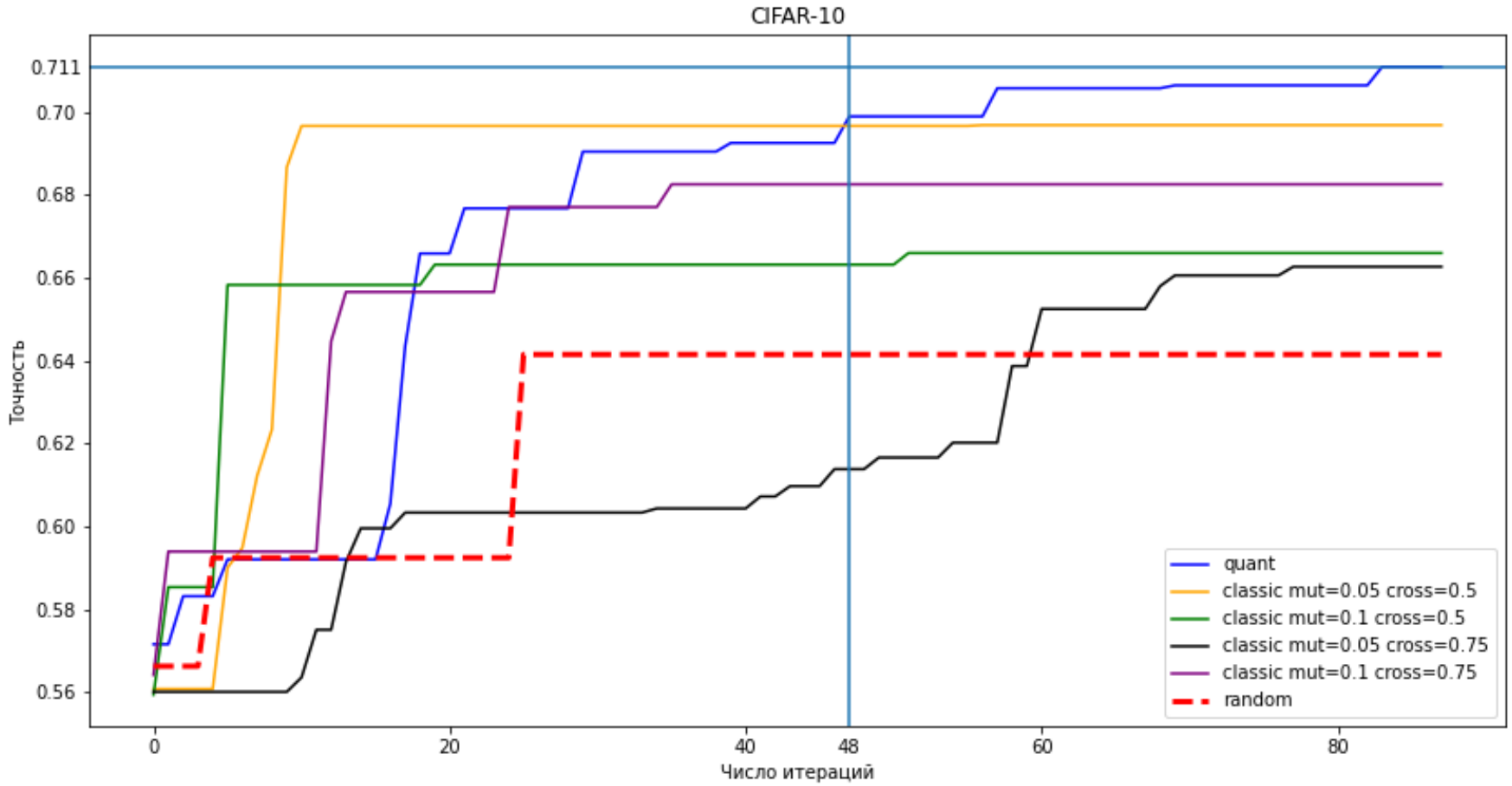
Аппаратная и программная платформы, использованные в вычислительном эксперименте.

- Эксперименты проводились на вычислительной системе Polus.
- 2 узла с использованием 2-ух ядер.
- Отдельный GPU для каждого потока.
- Обучение нейронных сетей проводилось с помощью библиотеки PyTorch для языка C++: libtorch версии v1.3.1. Функция с логикой процесса обучения нейронных сетей передаётся в ГА и КГА в качестве параметра (функция-приспособленности).
- Использование GPU для обучения нейросетей за счёт внутренних функций библиотеки libtorch.
- Время исполнения каждого из алгоритмов ≈ 3 часа.



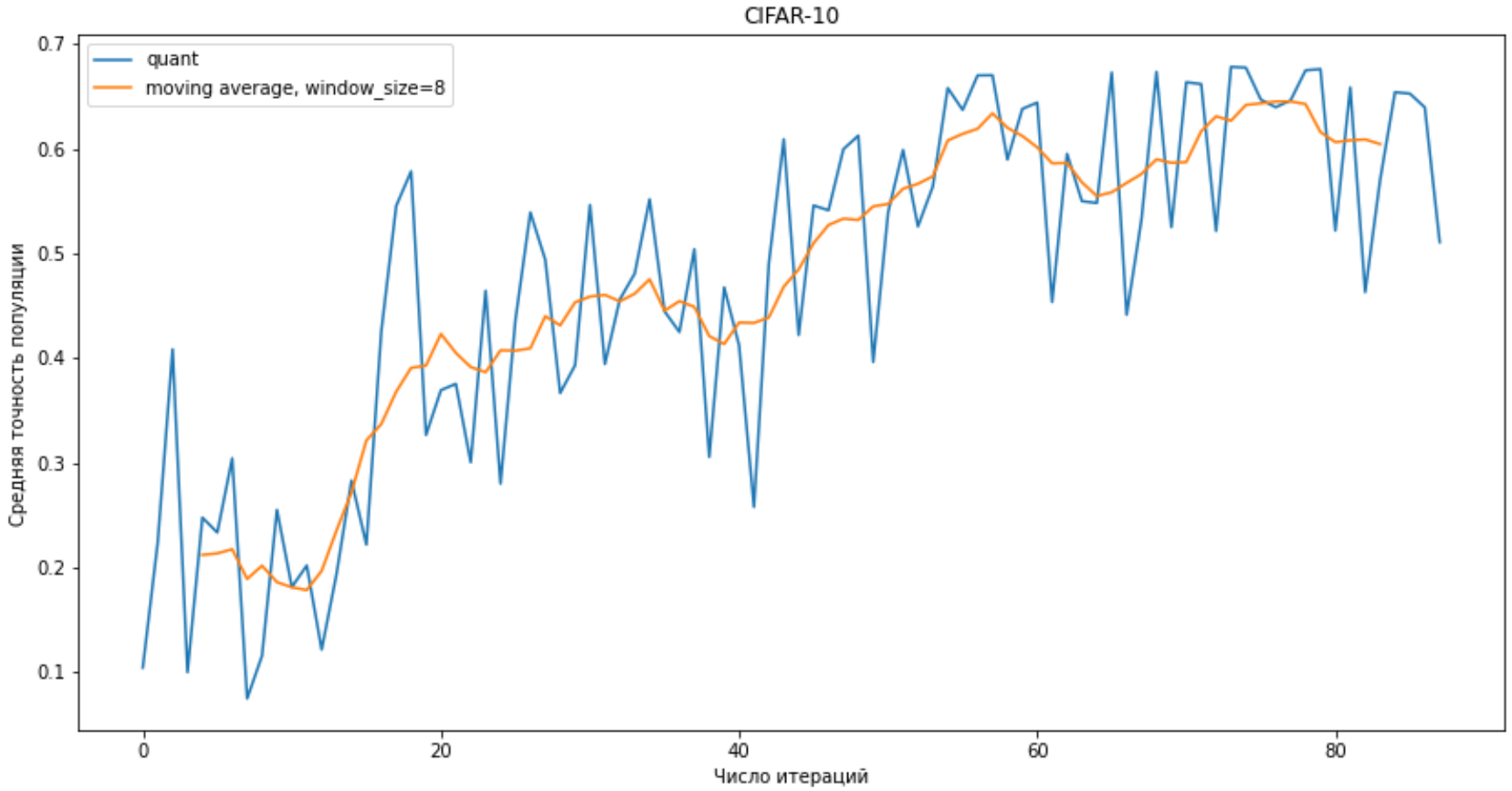
Результаты вычислительного эксперимента.

Максимальная достугнутая точность нейросетей, настроенных с использованием разработанных алгоритмов, на наборе данных CIFAR-10



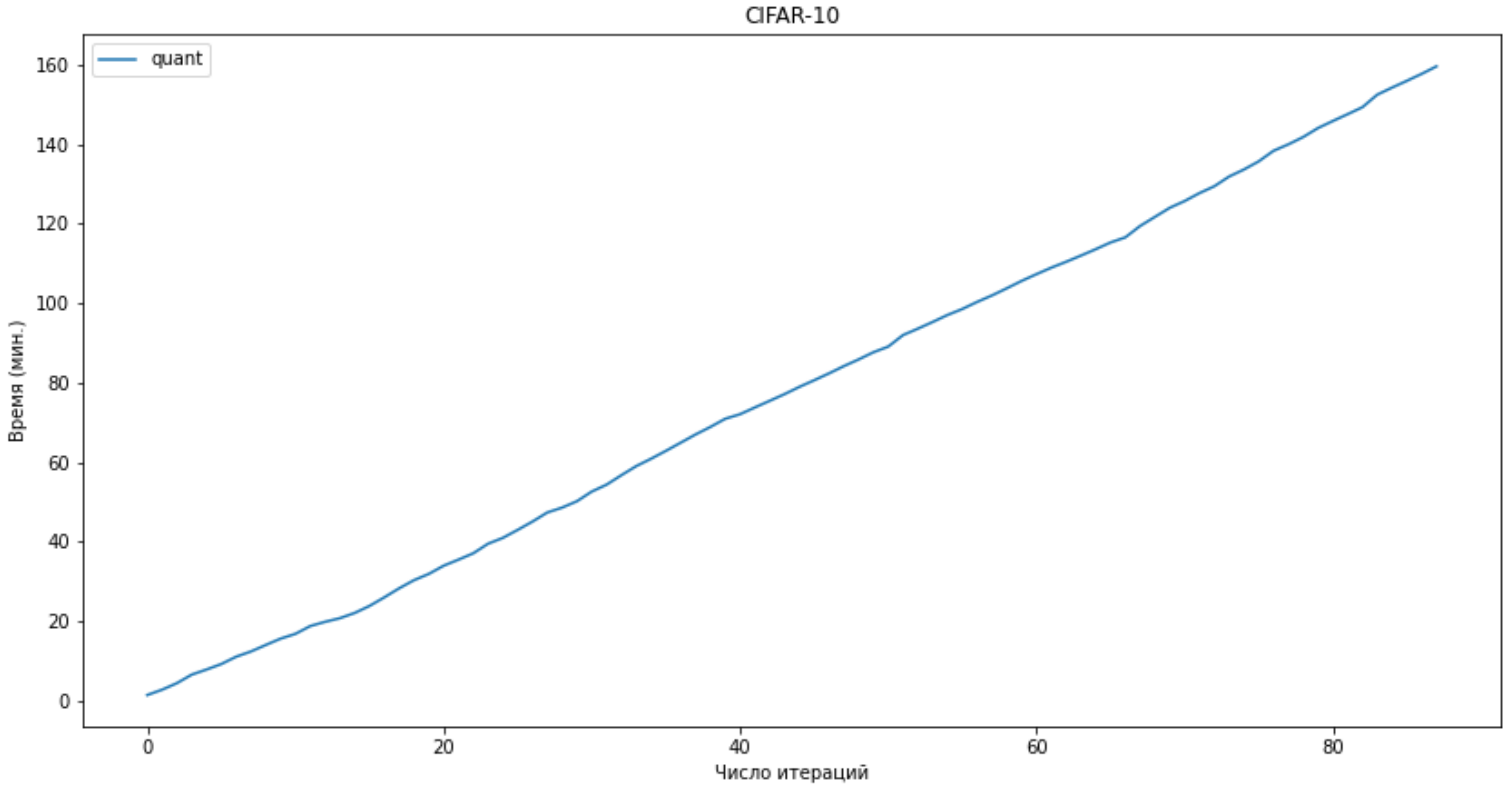
Результаты вычислительного эксперимента.

Средняя по популяции точность нейросетей, настроенных с использованием КГА, на наборе данных CIFAR-10



Результаты вычислительного эксперимента.

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



Polus:

- 2 узла
- На каждом 2 потока
- Отдельный GPU на каждый поток

Основные результаты

- Проведен обзор существующих подходов к решению задачи подбора архитектур нейронных сетей с использованием КГА.
- 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. 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. Вычислительный эксперимент выполнен на вычислительном кластере Polus. Время выполнения каждого из алгоритмов ≈ 3 часа.
- По результатам эксперимента показано преимущество использования КГА на примере решения задачи подбора архитектур свёрточных нейронных сетей для классификации изображений. Его результат превзошёл результаты ГА и метода случайного поиска. При этом в отличие от ГА, квантовый генетический алгоритм продолжает уточнять результат даже на последних итерациях.

Возможные пути дальнейшего исследования

- 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.
- Расширение набора решаемых задач оптимизации.