DSPA:

Вопросы применения цифровой обработки сигналов

№4

2025

СОДЕРЖАНИЕ

РАЗРБОТКА АЛГОРИТМА РАСПРЕДЕЛЕННОГО АНАЛИЗА ПРИ ОБРАБОТКЕ БОЛЬШИХ ДАННЫХ

Неманова Виктория Игоревна
МТУСИ, магистрант, Москва, Россия
vi.nemanova@gmail.com

Тремасова Лилия Андреевна
МТУСИ, ассистент кафедры СИТиС, к.т.н., Москва, Россия
l.a.tremasova@mtuci.ru

Калининский Даниил Сергеевич
МТУСИ, аспирант кафедры СИТиС, Москва, Россия
daniilblag28@mail.ru

Гадасин Денис Вадимович
МТУСИ, доцент кафедры СИТиС, к.т.н., Москва, Россия
dengadiplom@mail.ru

Аннотация
В данной статье представлена новая система распределенного анализа больших данных, которая обрабатывает данные на узлах сбора и центральном сервере. Это отличает её от существующих решений, в которых данные обрабатываются только на сервере. Предварительная обработка данных на этапе сбора значительно снижает количество информации, поступающей на сервер, что позволяет использовать обычные компьютеры для обработки. По двум основным параметрам – количеству обрабатываемых данных на центральном сервере и времени, необходимому для их обработки – результаты исследования показывают, что предложенный алгоритм превосходит традиционные распределенные методы.

Ключевые слова
Распределенный алгоритм, обработка информации, кластер, MapReduce, большие данные.

Введение

С увеличением популярности социальных сетей, облачных технологий, сенсорных сетей и систем доставки контента наблюдается значительный рост объема генерируемых данных, что стало основой для появления концепции «больших данных» [1-4]. Большие данные характеризуются тремя основными аспектами: объемом (количеством создаваемой информации), разнообразием (различными форматами данных, включая структурированные, полуструктурированные и неструктурированные) и скоростью (темпом их генерации и обработки).

В то же время, управление большими данными порождает ряд исследовательских вызовов, поскольку традиционные вычислительные системы и базы данных не имеют достаточной эффективности для обработки таких объемов информации. Для решения этих вызовов были разработаны алгоритмы распределенной обработки данных, которые осуществляют распределение вычислительных нагрузок между множеством менее мощных компьютеров, вместо полагания на один высокопроизводительный сервер. Данные алгоритмы основываются на концепции MapReduce, что позволяет значительно повысить эффективность обработки больших объемов информации.

Среди многих существующих распределенных алгоритмов выделяют параллельный алгоритм *IdeaGraph*, вероятностный латентно-семантический анализ (*PLSA*), алгоритм планирования с учетом локальности, алгоритм ближайшего соседа, алгоритм совместной фильтрации на основе элементов, алгоритм рекомендаций, выпуклый алгоритм оптимизация и параллельный двухпроходный *MDL (PTP-MDL)*, которые наиболее популярны в области обработки больших данных [5-10].

Описание разрабатываемого алгоритма

Распределенный алгоритм использует предложенную структуру больших данных и базируется на *MapReduce* на центральном сервере. Предлагаемый алгоритм масштабируем, динамичен, отказоустойчив и не зависит от приложения, работающего с большими данными [11-13, 21-27]. Алгоритм может использовать столько узлов, сколько необходимо. В зависимости от объема обрабатываемых данных он может использовать как последовательную, так и параллельную реализацию. Если объем данных превышает определенный порог, он работает в параллельном режиме; в противном случае он работает в последовательном режиме, чтобы снизить расходы на обработку, а также работает непрерывно, даже если один узел выходит из строя [14].

Общая архитектура предлагаемой структуры

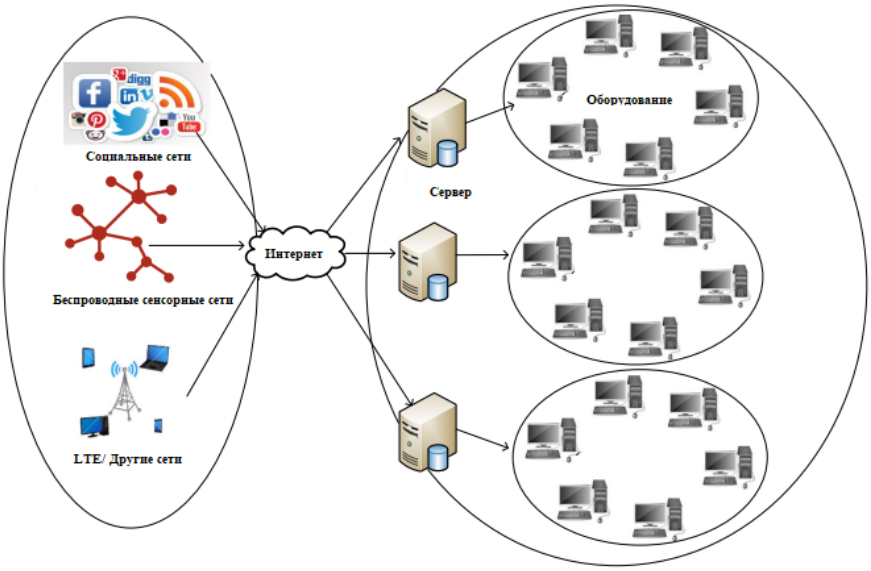


Рис. 1. Структура больших данных для разработки предлагаемого распределенного алгоритма

На рисунке 1 показана предлагаемая структура больших данных, которая используется в качестве основы для разработки распределенного алгоритма обработки больших данных. Распределенный алгоритм выполняет агрегацию данных на стороне сбора данных и анализ данных на стороне центрального сервера [15-17]. Эта структура имеет два конца сбора и обработки данных. На рисунке 1 показано, что левый конец осуществляет сбор данных и первоначальную обработку данных, тогда как правый конец получает уменьшенный объем данных с левого конца и осуществляет окончательную обработку данных с помощью обычных компьютеров. Обе стороны используют *MapReduce Hadoop* для распределенной обработки данных.

Предлагаемый подход работает путем создания кластера узлов при сборе данных (левый конец) и обработке данных (правый конец) структуры, а также разделения узлов каждого кластера на три группы и назначения им разных ролей [18]. Три группы узлов собой представляют:

1. Узлы сбора данных.
2. Группировка данных и сокращение узлов.
3. Главный узел, генерирующий окончательные агрегированные данные.

Рисунок 2 иллюстрирует группировку узлов кластера и назначение ролей каждой из этих групп. Группировка на основе приоритета используется там, где дается наивысший приоритет узлам с наибольшей энергией.

Первоначально настраивается кластер, состоящий из N узлов, расположенных в некоторой области A. Каждому узлу i (где i=1,2,...,N) назначаются координаты (x_i,y_i), для метрики расстояния используется евклидово расстояние:

$$d_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \quad (1)$$

В качестве главного узла (*MN*) выбирается узел, который находится ближе всего к центру области кластера, в данном случае это узел с координатами (x_{MN},y_{MN}). Это связано с тем, что сохранение *MN* в центре кластера уменьшает среднее расстояние передачи данных от всех других узлов к *MN*. Если узел *MN* – это узел *k*, то его выбор осуществляется по критерию минимизации суммарного расстояния до всех остальных узлов:

$$\min_k = \sum_{j \neq k} d_{kj} \tag{2}$$

Затем *MN* выбирает некоторое количество узлов в качестве узлов группировки/сокращения, которые выполняют операции группирования и сокращения, как определено в *MapReduce* [19]. *MN* выбирает несколько узлов в качестве группировки узлов, которые находятся очень близко к нему, а также имеет максимальную остаточную энергию по следующим условиям:

$$d_{MN,j} \leq d_{\max} \text{ и } E_j \leq E_{\min} \tag{3}$$

$\max_j E_j$ при условии $d_{MN,j} \leq d_{\max}$, где $d_{\max}$ – максимальное допустимое расстояние до *MN*; $E_{\min}$ – пороговая остаточная энергия.

Это приводит к снижению задержек при сквозной передаче данных и энергопотребления. *MN* также выбирает несколько узлов в качестве активных узлов сбора данных [20]. Эти узлы выбираются аналогично узлам группировки, но с дополнительным условием, что они должны быть в определенном состоянии готовности к сбору данных.

Остальные узлы остаются в режиме ожидания, сохраняя ресурсы для потенциальных задач в будущем. Эти узлы становятся любыми из этих трех типов, если кластер реконструируется из-за состояния энергии узлов сбора данных. Если узел *j* в качестве узла сбора данных выходит из строя (например, энергия падает ниже порога), то *MN* инициирует выбор нового узла:

$$j' arg \max_j E_j \text{ где } d_{MN,j} \leq d_{\max} \tag{4}$$

Рабочие нагрузки распределяются между оставшимися неактивными узлами по очереди для уравновешивания ресурсов. Это может быть выполнено путем:

$$W_i = \frac{T}{N_{active}} \tag{5}$$

где T − общая нагрузка.

Каждый узел *j* будет назначен нагрузку W_j в зависимости от своей текущей нагрузки и состояния.

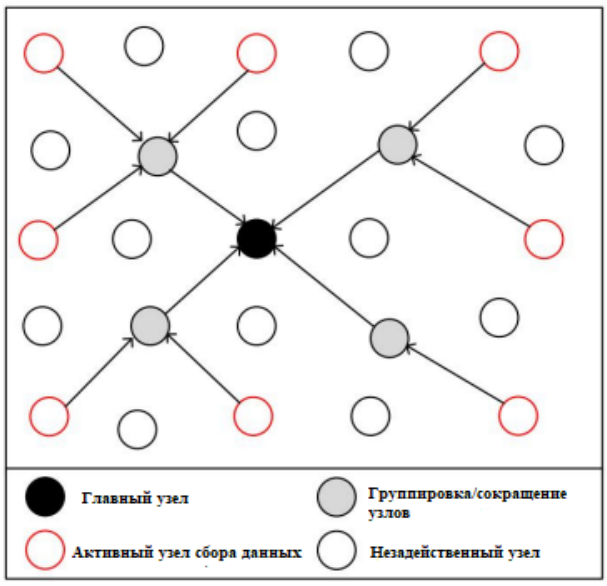


Рис. 2. Группировка узлов и распределение ролей

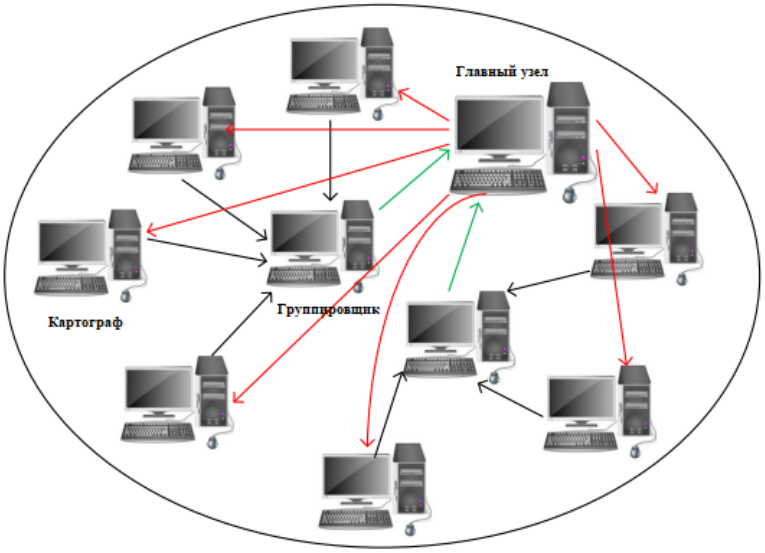


Рис. 3. Распределенная обработка больших данных на центральном сервере или главном узле

На рисунке 3 показано распределение процессов обработки данных по обычным компьютерам на центральном сервере, которые работают как *MN*, группировщики и картографы.

На рисунке 4 показано, как происходит отображение и уменьшение данных, собранных датчиками или другими компонентами в предложенной системе. Например, при сборе данных датчики регистрируют температуру, влажность, давление и другие параметры в течение определенного периода времени. Следующим шагом является операция сопоставления, которая включает в себя сортировку и удаление ненужных данных. Затем данные передаются узлу, который отвечает за группировку или сокращение. Эти группирующие датчики собирают данные от всех сборщиков и сохраняют одну копию идентичных пар (ключ или значение), полученных от различных датчиков. Кроме того, эти узлы выполняют математические функции для обработки данных, такие как сумма, максимум, минимум, среднее значение и медиана.

Затем датчики группировщика передают выходные данные на *MN*. *MN* генерирует окончательный результат после получения данных от всех группировщиков и передает через Интернет на центральный сервер.

Предложенный распределенный алгоритм имеет следующие характеристики:

1. Масштабируемость

Данный подход позволяет узлам интегрироваться в кластер или сеть (на этапе сбора данных) без необходимости изменения топологии сети. Узел отправляет сообщение «Join-Request» в сеть. Все *MN*, получившие этот запрос, отвечают сообщением «Join-Accept». После этого узел присоединяется к ближайшему кластеру, определяемому по уровню сигнала сообщения «Join-Accept».

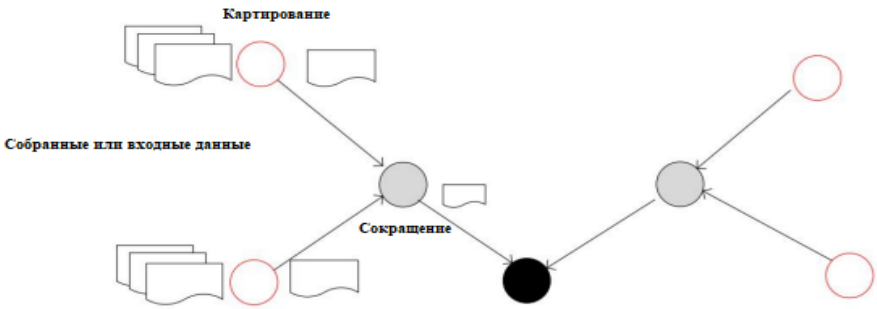


Рис. 4. Предлагаемый распределенный алгоритм обработки больших данных на основе *MapReduce*

Если в кластере отсутствуют сенсорные дыры или количество активных узлов для сбора данных достаточно, остальные узлы остаются на стадии простоя. В противном случае *MN* назначает узел по мере необходимости.

2. Динамичность

Даже в случае добавления новых узлов в сеть предложенный алгоритм автоматически подстраивается. В зависимости от количества данных, поступающих на центральный сервер, он может переключаться на последовательный или параллельный режим работы. Например, если объем данных более одного компьютера не может обрабатывать, алгоритм переводится в последовательный режим. Это позволяет центральному серверу передавать обработку данных одному устройству. Алгоритм переключается в параллельный режим, если объем данных значительно превышает этот предел. Это позволяет серверу распределять обработку данных между несколькими компьютерами одновременно.

3. Отказоустойчивость

Некоторые узлы остаются в режиме ожидания как на этапе сбора данных, так и на уровне центрального сервера. Эти бездействующие узлы резервируются для замены активных узлов. Узел *X* начинает функционировать только в случае, если он был назначен в качестве резервного для любого узла, вышедшего из строя, или если уровень энергии активного узла падает ниже установленного порога. Таким образом, если узел перестает работать, его функции переходят к бездействующему узлу *MN*. Поломка узла выявляется его непосредственным преемником в передаче данных. Например, непосредственными преемниками узлов сбора данных и группировщиков являются группировщики и *MN* соответственно. Если группировщик не получает данные от узла сбора данных в течение установленного времени, он считает, что узел сбора вышел из строя, и уведомляет об этом *MN*. Аналогично, *MN* принимает решение о выходе из строя группировщика, если не получает данные от него в течение заданного времени.

4. Параллельная и последовательная реализация

Предложенные алгоритмы могут функционировать на обычных компьютерах или узлах параллельно, особенно когда объем данных очень велик (что зависит от конкретного приложения и вычислительной мощности узлов). К примеру, все активные узлы сбора данных осуществляют свою работу параллельно для сбора или анализа информации, как показано на рисунке 2. Аналогично, узлы группировки действуют одновременно, с целью уменьшения объема данных. Пока группировщики выполняют процесс сокращения, узлы сбора данных продолжают сбор информации (то есть группировщики и сборщики работают синхронно). *MN* отвечает за координацию работ группировки и сборщиков данных. На стороне центральной обработки *MN* (то есть сервера) также распределяет задачи между несколькими рабочими станциями. Рабочие станции затем работают параллельно для сортировки и сопоставления данных, а полученные результаты с них отправляются группировщикам. Группировщики также функционируют одновременно. Если объем данных ниже определённого порога, *MN* отправляет информацию лишь на одну рабочую станцию. После этого эта станция выполняет операцию сопоставления и передаёт результаты группировщику для дальнейшей обработки. *MN* выбирает рабочую станцию и группировщик, основываясь на их вычислительных мощностях и объёме хранения, достаточных для выполнения текущей задачи.

Псевдокод

Рассмотрим практическую реализацию алгоритма на примере анализа лог-файлов веб-сервера.

Описание задачи

Современные компании, управляющие веб-сайтами, ежедневно сталкиваются с огромным количеством запросов от пользователей, что приводит к генерации больших объемов лог-файлов. Эти лог-файлы содержат ценную информацию о деятельности пользователей, включая посещённые страницы, время загрузки, возникшие ошибки, а также данные о производительности сайтов. Анализ логов позволяет командам выявлять популярное содержание, идентифицировать проблемы, такие как сбои на страницах или медленная загрузка, а также обнаруживать аномалии в поведении пользователей, что может сигнализировать о возможных попытках взлома или других ненормативных ситуациях.

Однако, с ростом масштаба операций возникает проблема обработки таких больших массивов данных. Передача всех логов на центральный сервер для последующего анализа может оказаться неэффективной из-за ограничений в пропускной способности канала и временных затрат, необходимых для передачи данных. Чтобы решить эту задачу, применим распределённый алгоритм, который позволит заранее обрабатывать данные на уровне веб-серверов, а затем осуществим финальную агрегацию и анализ уже на центральном сервере.

Архитектура решения

- Узлы сбора данных: Каждый веб-сервер будет выступать в роли узла сбора данных. Лог-файлы, генерируемые ими, будут храниться локально. На этом уровне можно выполнять различные предобработки, такие как фильтрация данных, идентификация ошибок, подсчёт количества запросов на каждую страницу. Это значительно снизит объём данных, которые затем необходимо отправлять на центральный сервер.
- Центральный сервер: Центральный сервер будет принимать агрегированные результаты анализа с каждого из узлов. Он будет отвечать за финальный анализ, объединение данных и получение сводной информации, такой как общие тренды, статистика по ошибкам или производительности. После этого полученные результаты могут быть использованы для создания отчётов, визуализации данных или для дальнейшего принятия решений о оптимизации работы веб-сайтов.

Описание парадигмы MapReduce

Этап 1: Локальная обработка на узлах

На первом этапе обработки данных веб-серверы, выступающие в роли узлов, выполняют следующие функции:

- Map: Каждый узел обрабатывает свои журналы (лог-файлы), извлекая из них ключевую информацию. Это может включать *URL*-адреса страниц, количество запросов к ним, а также коды ответов сервера. На этом этапе происходит разбиение данных на более мелкие блоки, подходящие для дальнейшей обработки.

Псевдокод:

```
def map_function(log_entry):
    # Разбираем строку лога
    url = extract_url(log_entry)
    status_code = extract_status_code(log_entry)
    return (url, 1), (status_code, 1)

# Обрабатываем все строки лога
mapped_data = [map_function(entry) for entry in log_file]
```

- Reduce: После завершения этапа Map данные агрегируются на каждом узле. Здесь осуществляется суммирование количества запросов на каждую отдельную страницу, подсчет частоты возникновения определенных кодов ошибок, а также формулирование статистики на основе собранной информации. Результаты локальной обработки подготавливаются для последующей передачи на следующий этап, что позволяет значительно сократить объем данных, передаваемых между узлами.

Псевдокод:

```
def reduce_function(mapped_data):
    url_counts = defaultdict(int)
    status_code_counts = defaultdict(int)

    # Агрегируем данные
    for (url, count), (status_code, status_count) in mapped_data:
        url_counts[url] += count
        status_code_counts[status_code] += status_count

    return url_counts, status_code_counts

reduced_data = reduce_function(mapped_data)
```

Этап 2: Передача агрегированных данных на центральный сервер

После того как данные были обработаны на каждом узле, они отправляются на центральный сервер для финальной обработки. Каждый узел передает информацию, включающую, например, списки из десяти самых запрашиваемых страниц и данные об ошибках 404, что позволяет получить общее представление о работе системы.

Этап 3: Финальная обработка на центральном сервере

Центральный сервер объединяет данные от всех узлов и выполняет финальную агрегацию.

- Map: На этом этапе сервер собирает входящие данные от всех узлов, формируя единую картину запросов и ошибок.
- Reduce: Затем система суммирует количество запросов для каждого уникального *URL*, создавая общую статистику по всем узлам. Эта информация может быть использована для дальнейшего анализа, оптимизации производительности и выявления проблем в системе.

Псевдокод:

```
def central_reduce_function(all_node_data):
    global_url_counts = defaultdict(int)
    global_status_code_counts = defaultdict(int)

    for node_data in all_node_data:
        url_counts, status_code_counts = node_data
        for url, count in url_counts.items():
            global_url_counts[url] += count
        for status_code, count in status_code_counts.items():
            global_status_code_counts[status_code] += count

    return global_url_counts, global_status_code_counts

final_result = central_reduce_function(reduced_data_from_all_nodes)
```

Результаты

После завершения финальной обработки данные на центральном сервере могут быть представлены в виде аналитических отчетов, включая, но не ограничиваясь, следующими показателями:

- Топ-10 страниц с наибольшим количеством запросов на всех веб-сайтах, что позволит выявить наиболее популярное содержимое и потенциал для дальнейшего продвижения.
- Подробная статистика по ошибкам, в том числе общее количество ошибок 404 и других типов ошибок, что поможет в оптимизации веб-ресурсов и улучшении пользовательского опыта.
- Выявление аномальных паттернов в запросах, которые могут указывать на возможные проблемы или угрозы, требующие немедленного внимания и анализа.
- Анализ временных тенденций в запросах, что может помочь в планировании будущих обновлений контента и улучшении маркетинговых стратегий.
- Сравнительный анализ производительности различных веб-сайтов, что позволит выявить сильные и слабые стороны в подходах к управлению контентом и взаимодействию с пользователями.

Эти данные могут быть полезны для принятия обоснованных решений и стратегического планирования на основе поведения пользователей и технических характеристик веб-сайтов.

Оценка эффективности

Проведем оценку производительности предложенного распределенного алгоритма для обработки больших данных. Оценка основана на анализе объема данных, обрабатываемых на обычных компьютерах, времени обработки данных, а также на результатах моделирования.

Анализ производительности

Предположим, что размер файла или пакета данных, собираемого узлом, равен d_{size1}. Также предположим, что количество кластеров обозначено как n_{clust}, количество узлов сбора данных в одном кластере – как n_{cnode}, а количество узлов группировки в кластере – как n_{gnode}, при этом $n_{gnode} \le n_{cnode}$.

Среднее количество пакетов данных, собираемых узлом за интервал времени *T*, равно p. Количество пакетов, передаваемых узлом на узел группировки после операции отображения, равно (*p* - *q*), где $q \ge 0$.

Таким образом, общий объем данных, передаваемых узлами сбора данных в кластере, можно выразить следующим образом:

$$D_{tx1} \le n_{cnode} \times d_{size1} \times (p - q) \quad (6)$$

Предположим, что каждый узел группировки получает примерно одинаковый объем данных (т.е. данные равномерно распределяются между узлами группировки). Тогда объем данных, получаемых каждым узлом группировки, будет равен:

$$D_{rx-gr} \le \frac{(n_{cnode} \times d_{size1} \times (p-q))}{n_{gnode}} \tag{7}$$

Предположим также, что каждый узел группировки передает данные размером d_{size2} на главный узел (*MN*) после выполнения операции группировки или сокращения, где $d_{size2} > d_{size1}$.

Тогда общий объем данных, получаемых главным узлом (*MN*), будет равен:

$$d_{size3} = n_{gnode} \times d_{size2} \tag{9}$$

Таким образом, сложность данных алгоритма на стороне сбора данных составляет *O(d)*.

Аналогично, если предположить, что количество главных узлов (*MN*), которые передают данные на центральный сервер для дальнейшей обработки, равно m, общий объем данных, получаемых сервером, будет равен:

$$d_{size4} = m \times d_{size3} \tag{10}$$

Если количество обычных компьютеров, распределяющих нагрузку (данные), равно q, то объем данных, получаемых каждым компьютером, будет:

$$d_{size5} = \frac{d_{size4}}{q} = \frac{m \times d_{size3}}{q} \tag{11}$$

Таким образом, сложность данных алгоритма на стороне центрального сервера или обычных компьютеров также составляет *O(d)*.

Настройка моделирования и результаты

Для оценки производительности предложенного распределенного алгоритма обработки больших данных было проведено моделирование, в котором рассматривался объем данных, получаемых сервером на этапе центральной обработки данных, а также время обработки данных. Модель моделирования была реализована с использованием языка программирования C и включала четыре кластера узлов на стороне сбора данных. В каждом кластере было максимум 60 узлов сбора данных и 20 узлов группировки (или сокращения). Аналогично, на стороне сервера было максимум 60 обычных компьютеров.

Большинство из них выполняли функции маппера, а несколько – редьюсера. Моделирование проводилось в течение 32 минут. Узлы сбора данных были настроены на сбор данных каждые 2 минуты (т.е. сбор, сортировка и отображение). После сбора данных они передавались узлам группировки, которые выполняли операцию сокращения и передавали результат главному узлу (*MN*). Главный узел на стороне сбора данных затем передавал данные на сервер (т.е. *MN* на стороне центральной обработки) через Интернет. Сервер распределял данные между различным числом мапперов и редьюсеров (максимум 32 маппера и 12 редьюсеров). Время обработки измерялось на стороне центрального сервера и определялось как разница во времени между передачей данных от *MN* к мапперам и редьюсерам и получением обработанных данных на сервере для дальнейшего хранения.

Результаты моделирования показаны на рисунках 5, 6, 7 и 8. Рисунки 5 и 6 показывают объем данных, которые получает *MN* на стороне сбора данных при изменении количества узлов сбора данных и узлов группировки, соответственно. Это также объем данных, который *MN* передает на центральный сервер для обработки. Было обнаружено, что объем данных, обрабатываемых предложенным подходом, примерно в 4-5 раз меньше, чем в существующих подходах, которые не учитывают обработку данных на стороне сбора данных. Поскольку данные обрабатываются на обоих концах структуры с использованием парадигмы MapReduce, ожидается, что объем данных как на *MN* на стороне сбора данных, так и на центральном сервере/обычных компьютерах будет значительно меньше.

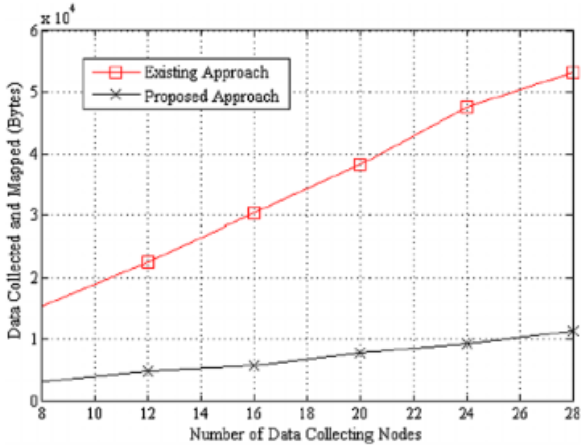


Рис. 5. Размер данных, обрабатываемых главным узлом (MN) на стороне сбора данных, при изменении количества узлов сбора данных

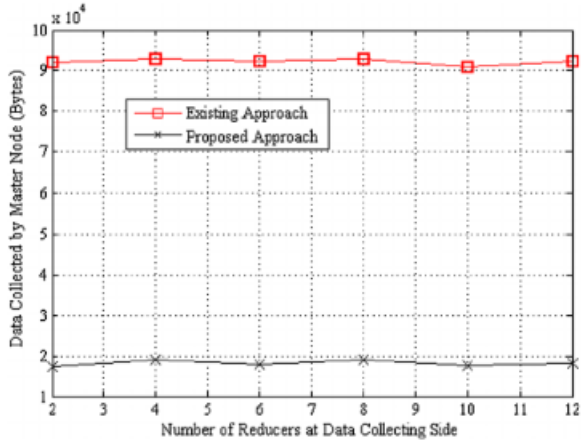


Рис. 6. Размер данных, собираемых главным узлом (MN) на стороне сбора данных, при изменении количества узлов группировки

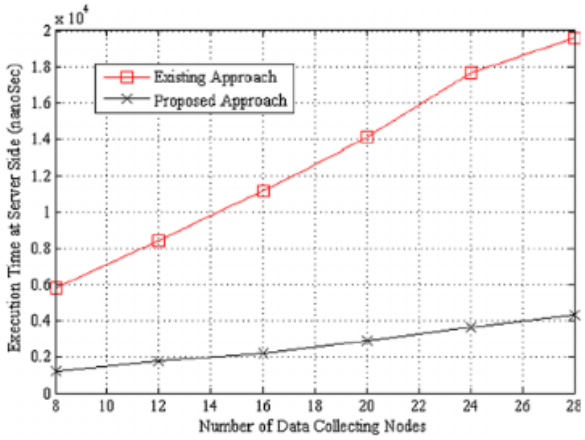


Рис. 7. Время обработки данных на обычных компьютерах при изменении количества узлов сбора данных

Рисунки 7 и 8 показывают время обработки данных на центральном сервере, который получает частично обработанные данные от MN на стороне сбора данных. Мы обнаружили, что время обработки данных в предложенном подходе примерно в пять раз меньше, чем в традиционном распределенном алгоритме для больших данных. Это объясняется тем, что обработка распределена как на стороне сбора данных, так и на стороне обработки данных. Таким образом, ожидается, что нагрузка на обработку, а также время обработки на обычных компьютерах будут значительно сокращены.

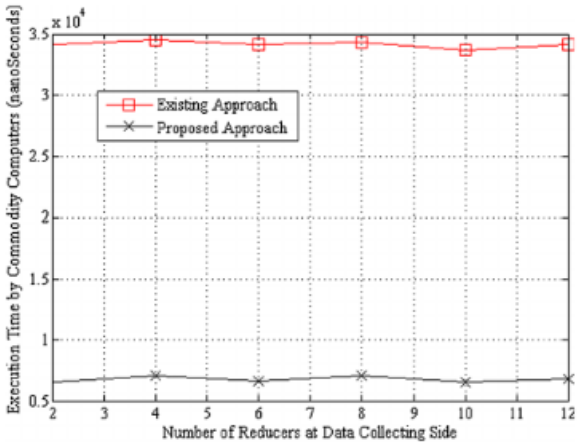


Рис. 8. Сравнение времени выполнения обработки данных на обычных компьютерах при изменении количества редьюсеров

Заключение

Таким образом, разработанный алгоритм обеспечивает значительное повышение эффективности обработки больших данных благодаря оптимальному использованию ресурсов и сокращению времени отклика системы. Его адаптивная природа позволяет динамически переключаться между параллельной и последовательной обработкой в зависимости от текущей ситуации, что способствует снижению накладных расходов и увеличению производительности.

Кроме того, алгоритм обладает отказоустойчивыми свойствами, что является критически важным для современных распределенных систем, где сбои узлов могут повлиять на общую эффективность. Возможность обработки как на этапе сбора данных, так и на центральном сервере позволяет равномерно распределять нагрузку и снижать риск перегрузки сервера.

В результате, алгоритм не только улучшает вычислительную производительность, но и обеспечивает гибкость интеграции с различными приложениями, что делает его универсальным инструментом для работы с большими данными. Это открывает новые перспективы для создания более сложных и масштабируемых систем анализа данных, способных справляться с растущими объемами информации в реальном времени. В конечном итоге, предложенный подход может значительно повысить эффективность бизнес-процессов и улучшить качество принимаемых решений на основе анализа данных.

Литература

1. *Wu C., Birch D., Silva D., Lee C.-H., Tsinalis O., Guo Y.* Concinnity: a generic platform for big sensor data applications // IEEE In Cloud Comput. 2014, pp. 42–50.

2. *Michael K., Miller K.W.* Big data: new opportunities and new challenges // Computer 46(6). 2013, pp. 22-24.

3. *Гадасин Д.В., Шведов А.В.* Применение транспортной задачи для балансировки нагрузки в условиях нечеткости исходных данных // T-Comm: Телекоммуникации и транспорт. 2024. Т. 18, № 1. С. 13-20. DOI 10.36724/2072-8735-2024-18-1-13-20. EDN WKNPIX

4. *Liang Z., Li W., Li Y.* A parallel probabilistic latent semantic analysis method on MapReduce platform // 2013 IEEE International Conference on Information and Automation (ICIA), Yinchuan, 2013, pp. 1017-1022.

5. *Panteleeva K.A., Shevelev S.V., Pervukhina A.A., Gadasin D.V.* Accident Process Management System in the IT-Landscape as a Means of Ensuring the Structural Reliability of the Organization // 2024 Systems of Signal Synchronization, Generating and Processing in Telecommunications (SYNCHROINFO), Vyborg, Russian Federation, 2024, pp. 1-7, doi: 10.1109/SYNCHROINFO61835.2024.10617446.

6. *Savin V.A., Shevelev S.V., Shulpina P.D., Gadasin D.V.* Development of a Model of Distributed Computing Systems // 2024 Systems of Signals Generating and Processing in the Field of on Board Communications, Moscow, Russian Federation, 2024, pp. 1-8, doi: 10.1109/IEEECONF60226.2024.10496732.

7. *Gadasin D.V., Shvedov A.V., Egorova Y.D., Shaidullina I.R.* Application of the Method of Majority Coding to Determine the Optimal Route for Data Transmission in the Network // 2023 Systems of Signals Generating and Processing in the Field of on Board Communications, Moscow, Russian Federation, 2023, pp. 1-6, doi: 10.1109/IEEECONF56737.2023.10092067.

8. *Krishnan N., Baron D*. A universal parallel two-pass MDL context tree compression algorithm // IEEE J. Sel. Top. Signal Process. 9(4), 2015, pp. 741-748.

9. *Melkova E.K., Shvedov A.V., Korovushkina V.M., Gadasin D.V*. Cluster Implementation Based on the Belonging Function // 2023 Systems of Signal Synchronization, Generating and Processing in Telecommunications (SYNCHROINFO, Pskov, Russian Federation, 2023, pp. 1-6, doi: 10.1109/SYNCHROINFO57872.2023.10178611

10. *Tremasova L.A., Korovushkina V.M., Panteleeva K.A., Gadasin D.V.* Distributed Information System Software Code Reliability Evaluation // 2024 Wave Electronics and its Application in Information and Telecommunication Systems (WECONF), St. Petersburg, Russian Federation, 2024, pp. 1-5, doi: 10.1109/WECONF61770.2024.10564614.

11. *Золотарева П.Ю., Гадасин Д.В., Маклачков К.А*. Методы обработки информации в распределенных информационных системах // Тенденции развития Интернет и цифровой экономики : Труды VI Международной научно-практической конференции, Симферополь-Алушта, 01-03 июня 2023 года. Симферополь: ИП Зуева, 2023. С. 187-189. EDN LGONZK

12. *Гадасин Д.В., Бессолицын А.Д*. Виды и методы структурирования данных из различных информационных систем: анализ и применение // Актуальные проблемы и перспективы развития экономики, Симферополь – Гурзуф, 12-14 октября 2023 года. Симферополь: ИП Зуева Т. В., 2023. С. 202-204. EDN UGZRXL

13. *Гадасин Д.В., Пантелеева К.А., Маклачков К.А*. Разработка единой точки входа сообщений о пользовательском негативном опыте взаимодействия с web-сервисами // Искусственный интеллект в автоматизированных системах управления и обработки данных : Сборник статей II Всероссийской научной конференции. В 5-ти томах, Москва, 27–28 апреля 2023 г. М.: Издательский дом КДУ, "Добросвет", 2024. С. 413-417. EDN ADRGFV

14. *Гадасин Д.В., Вакурин И.С., Тремасова Л.А*. Алгоритм распределения данных между системами хранения на основе свойства самоподобия // Электросвязь. 2024. № 4. С. 44-50. DOI 10.34832/ELSV.2024.53.4.007. EDN BRSLCL

15. *Гадасин Д.В., Шведов А.В., Кузин И.А*. Трехмерная реконструкции объекта по одному изображению с использованием глубоких свёрточных нейронных сетей // T-Comm: Телекоммуникации и транспорт. 2022. Т. 16, № 7. С. 29-35. DOI 10.36724/2072-8735-2022-16-7-29-35. EDN YTLCNW

16. *Гадасин Д.В., Золотарева П.Ю., Тремасова Л.А.* Влияние кластеризации при обработке сырых данных // Системы синхронизации, формирования и обработки сигналов. 2024. Т. 15, № 3. С. 10-19. EDN JQIPHX

17. *Яковенко Н.В., Гадасин Д.В., Коцич Л*. Повышение точности коэффициента влияния ошибок в информационных системах с применением метода обратного распространения ошибки // Системы синхронизации, формирования и обработки сигналов. 2024. Т. 15, № 4. С. 35-42. EDN CMFVNH

18. *Гадасин Д.В., Андриянова А.К., Тремасова Л.А.* Определение нечеткости поискового запроса через множество аксиом объекта // Технологии информационного общества : Сборник трудов XVII Международной отраслевой научно-технической конференции, Москва, 02-03 марта 2023 г. М.: Издательский дом Медиа Паблишер, 2023. С. 132-134. EDN DWJUTE

19. *Гадасин Д.В., Михайлов М.Р., Чернышов Д.В.* Определение алгоритма структурирования текстовых данных // REDS: Телекоммуникационные устройства и системы. 2024. Т. 14, № 1. С. 4-11. EDN GLAEQF

20. *Гадасин Д.В., Первухина А.А.* Группировка узлов передачи данных с использованием метода к-средних для создания кластеров // Теория и практика экономики и предпринимательства: Труды XXI Международной научно-практической конференции, Симферополь – Гурзуф, 18-20 апреля 2024 г. Симферополь: ИП Зуева Т. В., 2024. С. 233-236. EDN IAHHLX

21. *Shvedov A.V., Gadasin D.V., Alyoshintsev A.V.* Segment routing in data transmission networks // T-Comm. 2022. Vol. 16. No. 5, pp. 56-62. DOI: 10.36724/2072-8735-2022-16-5-56-62 EDN: VAYLJQ

22. *Назаров М.Д., Шведов А.В.* Корреляция атрибутов соглашения об уровне обслуживания с основными параметрами QoS в корпоративных сетях // Телекоммуникации и информационные технологии. 2020. Т. 7. № 2. С. 73-79. EDN: VQHDTJ

23. *Kalmykov N.S., Dokuchaev V.A.* Segment routing as a basis for software defined network // T-Comm. 2021. Т. 15. № 7. С. 50-54. EDN: LYVZCV

24. *Dokuchaev V.A., Maklachkova V.V., Statev V.Yu.* Classification of personal data security threats in information systems // T-Comm. 2020. Т. 14. № 1. С. 56-60. EDN: QOGYHH

25. *Докучаев В.А., Маклачкова В.В., Статьев В.Ю.* Цифровизация субъекта персональных данных // T-Comm: Телекоммуникации и транспорт. 2020. Т. 14. № 6. С. 27-32. EDN: XVWYJP

26. *Pavlov S.V., Dokuchaev V.A., Mytenkov S.S.* Model of a fuzzy dynamic decision support system // T-Comm. 2020. Т. 14. № 9. С. 43-47. EDN: VYFNLB

27. Кузин И.А., Гадасин Д.В. Модель контейнера данных для минимизации трафика при передаче субъективных характеристик объектов на изображении трехмерной сцены // Телекоммуникации и информационные технологии. 2021. Т. 8. № 2. С. 96-100. EDN: TYFFBH

ОБЕСПЕЧЕНИЕ ЗАЩИТЫ СИСТЕМЫ МАШИННОГО ОБУЧЕНИЯ СОВРЕМЕННЫМИ КРИПТОГРАФИЧЕСКИМИ МЕТОДАМИ

Орлова Александра Сергеевна
МТУСИ, студент, Москва, Россия
Orlovaalex1612@gmail.com

Панков Константин Николаевич
МТУСИ, заведующий кафедрой «Теория вероятностей и прикладная математика», к.ф.-м.н., доцент, Москва, Россия
pankov_kn@mtuci.ru

Аннотация
В данной работе рассматриваются современные опасности и методы защиты систем машинного обучения. Современные методы, такие как дифференциальная приватность, многосторонние вычисления и гомоморфное шифрование, уделяют особое внимание защите данных от утечек, атак на входные параметры и несанкционированного копирования моделей. В статье рассматриваются основные принципы работы этих методов, их применение к различным сценариям управления производительностью (МО) и их влияние на точность и производительность моделей. Проведен сравнительный анализ методов, представленных в научных исследованиях, чтобы определить их преимущества, недостатки и возможности в области обеспечения безопасности МО.

Ключевые слова
Машинное обучение, информационная безопасность, криптография, гомоморфное шифрование, многосторонние вычисления, дифференциальная приватность, защита данных, атаки на модели, утечки информации

Введение

В современном мире системы машинного обучения (далее — МО) как способа создания и обучения искусственного интеллекта становятся неотъемлемой частью таких сфер, как здравоохранение, финансы, транспорт и кибербезопасность. Их внедрение приносит многочисленные преимущества, однако одновременно порождает новые вызовы, связанные с обеспечением безопасности. Среди основных угроз можно выделить манипуляции с входными данными, утечку конфиденциальной информации и кражу моделей. Эти риски подрывают надёжность и доверие к технологиям МО.

Обеспечение безопасности систем МО занимает ключевое место в деятельности специалистов, занимающихся как практическими задачами, так и теоретическими исследованиями. Традиционные методы информационной безопасности по мнению ряда специалистов оказались недостаточно эффективными для защиты таких систем, что подчёркивает необходимость использования современных криптографических технологий. Среди наиболее перспективных решений выделяются дифференциальная приватность, многосторонние вычисления (MPC) и гомоморфное шифрование, которые позволяют защитить данные и модели от несанкционированного доступа и атак.

Научные статьи и исследования уделяют особое внимание разработке методов, направленных на минимизацию рисков и повышение устойчивости систем МО. Настоящая работа посвящена детальному анализу ключевых угроз, связанных с применением технологий машинного обучения, а также рассмотрению криптографических подходов, направленных на их предотвращение. Кроме того, проведён сравнительный анализ различных подходов, что позволило выявить их сильные и слабые стороны, а также определить перспективные направления для дальнейшего развития в области защиты систем машинного обучения.

Основные угрозы безопасности для систем машинного обучения

С ростом использования систем МО появляются новые уязвимости и угрозы. Например, злоумышленники могут нарушить работу модели, внося незначительные изменения в данные, что приводит к неверным предсказаниям. Это особенно опасно в случае автономных автомобилей, где изменение дорожного знака может привести к неправильной интерпретации [1]. Для защиты от таких атак

применяются устойчивые архитектуры моделей и противоположное обучение – метод, которые включает обучение модели на искусственно созданных атакующих данных.

Другой серьёзной угрозой являются утечки данных. Модели МО могут сохранять информацию об обучающих данных, что делает их уязвимыми для восстановления злоумышленниками. Например, атака Membership Inference позволяет определить, принадлежит ли конкретное наблюдение обучающему набору модели. Даже такие методы защиты, как дифференциальная приватность, требуют тщательной настройки, чтобы избежать утечек. Особенно уязвимы системы, работающие в облачной среде, где данные передаются между устройствами и серверами.

Кроме того, существует риск кражи моделей. Злоумышленники могут создать копию модели, отправляя многочисленные запросы к её API и анализируя полученные результаты. Это угрожает интеллектуальной собственности компаний, разрабатывающих сложные и дорогостоящие модели. Для защиты используются такие меры, как ограничение количества запросов, применение шифрования и добавление шума к ответам API, хотя эти методы могут снижать производительность системы.

Ещё одной угрозой являются атаки через побочные каналы, при которых извлечение информации осуществляется через анализ побочных характеристик системы, таких как потребление энергии, время выполнения операций или электромагнитные излучения. Например, в федеративном обучении возможны утечки через обновления градиентов, передаваемых между участниками. Для защиты применяются рандомизация вычислений, шифрование передаваемых данных и специальные аппаратные решения.

Важным аспектом остаётся обеспечение целостности и аутентичности данных, используемых для обучения. Нарушение этих свойств может привести к созданию предвзятых моделей или появлению неожиданных уязвимостей в работе систем МО.

Эти угрозы подчёркивают необходимость внедрения новых инструментов для защиты моделей и данных, таких как криптографические методы, регуляризация и мониторинг поведения моделей в реальном времени [2].

Современные криптографические методы защиты

Криптографические методы обеспечивают защиту данных и моделей от широкого диапазона угроз. Среди наиболее востребованных методов специалисты выделяют гомоморфное шифрование, многосторонние вычисления (MPC) дифференциальная приватность и системы распределенного реестра или блокчейн системы. Рассмотрим их более подробно.

Гомоморфное шифрование (далее – ГШ) позволяет выполнять вычисления над зашифрованными данными без их расшифровки, что делает его незаменимым в облачных вычислениях и федеративном обучении. Это решение поддерживает сложные операции, такие как матричные вычисления, широко применяемые при обучении нейронных сетей. Благодаря этому подходу конфиденциальность информации сохраняется даже при взаимодействии с удалёнными серверами [3].

ГШ представляет собой мощный инструмент для обеспечения конфиденциальности данных. Основная идея этого метода заключается в выполнении операций над зашифрованными данными без их раскрытия.

К преимуществам ГШ относятся, в соответствии с [4] и [5]:

- Конфиденциальность, под которой мы понимаем то, что данные остаются защищёнными на всех этапах обработки. Например, финансовые организации могут использовать ГШ для анализа транзакций клиентов, исключая доступ третьих сторон к деталям операций.
- Гибкость, под которой мы понимаем то, что поддержка сложных операций, таких как сложение и умножение в полностью гомоморфном случае, делает этот метод незаменимым для машинного обучения, где матричные вычисления играют ключевую роль.
- Применимость в облачных вычислениях, под которой мы понимаем то, что пользователи могут безопасно делегировать обработку данных сторонним провайдерам. Это особенно актуально для медицинских исследований, таких как анализ геномных данных.

К числу ограничений ГШ можно [4, 5] отнести:

- Высокая вычислительная сложность, под которой мы понимаем то, что операции с зашифрованными данными требуют значительных ресурсов. Современные исследования направлены на оп-

тимизацию алгоритмов, таких как схемы CKKS, которые поддерживают операции с плавающей точкой.

- Ограниченная масштабируемость: Обработка больших объёмов данных, особенно в реальных сценариях, таких как обучение глубоких нейронных сетей, остаётся серьёзной задачей для методов ГШ.

Отметим, что современные методы полностью ГШ основаны на тех же математических задачах, что и ряд постквантовых алгоритмов [6]. Таким образом, данный метод защиты систем МО является актуальным в условиях квантового вызова [7].

Многосторонние вычисления (английская аббревиатура – MPC, данное обозначение будем использовать далее) предоставляют возможность нескольким сторонам выполнять совместные вычисления, не раскрывая друг другу свои данные. Это особенно важно для приложений в медицинской и финансовой сферах, где требуется высокая степень защиты. Например, медицинские организации могут обучать модели на распределённых данных пациентов, не нарушая конфиденциальности информации [4].

MPC позволяет участникам совместно выполнять вычисления без раскрытия исходных данных. Это делает метод особенно ценным для федеративного обучения, где данные остаются на устройствах пользователей.

К преимуществам MPC обычно относят:

- Полную конфиденциальность, под которой мы понимаем то, что каждый участник получает только общий результат. Например, несколько больниц могут обучать общую модель диагностики заболеваний, не раскрывая данные пациентов друг другу [4].
- Устойчивость к утечкам, под которой мы понимаем то, что отсутствие необходимости передачи данных снижает риск компрометации. Это делает MPC идеальным решением для финансового анализа, где требуется объединение данных из нескольких банков [8].

Примеры применения MPC в настоящее время можно наблюдать в разных сферах. К примеру, в здравоохранении может быть проведено совместное обучение моделей диагностики заболеваний на данных из разных клиник. Примером является обучение модели для выявления онкологических заболеваний, где данные пациентов остаются локальными [11]. В сфере финансов можно проводить анализ транзакций между банками без раскрытия клиентских данных. Это позволяет выявлять подозрительные транзакции, сохраняя конфиденциальность информации [4, 9]. При обработке данных в системах интернета вещей (IoT) MPC используется для объединения данных с сенсоров, расположенных в разных географических точках, для анализа климатических изменений [4, 11].

К числу ограничений MPC в соответствии с [4, 10, 11] можно отнести

- Высокую вычислительную сложность и сложность реализации протоколов, под которой мы понимаем то, что, несмотря на прогресс, такие задачи, как безопасное вычисление градиентов, требуют значительных усилий.
- Масштабируемость, которая остаётся вызовом, особенно при увеличении числа участников и объёма данных.

Дифференциальная приватность (далее – DP) минимизирует риск утечек, добавляя случайный шум к результатам анализа или предсказаниям модели. Этот подход активно применяется для защиты пользовательских данных в рекомендательных системах и других задачах, связанных с обработкой информации пользователей. Например, анализ предпочтений пользователей в интернет-магазинах может проводиться с сохранением их анонимности [11].

DP минимизирует утечку информации, добавляя контролируемый шум в процессе обработки данных. Принципы DP используются как в централизованных системах, так и в федеративном обучении.

К числу примеров использования DP можно отнести:

- Обучение моделей. В этом случае шум добавляется к градиентам для предотвращения восстановления исходных данных. В задачах, связанных с медициной, это позволяет анализировать данные пациентов без риска утечек [11].
- Агрегацию данных. В этом случае происходит защита статистики, извлекаемой из больших массивов данных. Например, при обработке данных опросов DP гарантирует, что результаты анализа не раскроют информацию о конкретных респондентах [12].

- Рекомендательные системы. В этом случае происходит использование DP для персонализации без риска утечек пользовательских предпочтений, что делает её важным инструментом для e-commerce [11, 12].

Системы распределенного реестра или блокчейн [13] позволяет обеспечить неизменность данных и отслеживать изменения моделей, что укрепляет доверие к их целостности. Этот метод хранит записи об обновлениях моделей и транзакциях, делая их прозрачными и защищёнными от несанкционированного доступа [14, 15].

Блокчейн-технология обеспечивает неизменность и прозрачность обработки данных. В контексте МО она используется для записи изменений в моделях и проверки их подлинности. Отметим, что блокчейн системы могут применяться в сочетании с квантовыми методами защиты информации [16].

К числу способов применения систем распределенных реестров можно отнести защиту моделей, а именно фиксирование версии модели и контроль доступа. Например, блокчейн может использоваться для хранения истории изменений моделей в финансовых системах, что позволяет отслеживать все модификации и предотвращать их подделку [17]. Также применение этих технологий возможно в доверенных вычислениях, а именно возможно использование смарт-контрактов для автоматизации обработки данных. Это актуально для управления цепочками поставок, где требуется защита данных о транзакциях [18].

Эти способы использования является одновременно и преимуществами блокчейна.

Также к числу примеров практического применения данной технологии относятся такие сферы как:

- Здравоохранение, в котором возможна фиксация истории изменений моделей диагностики, что позволяет отслеживать их точность и выявлять возможные ошибки.
- Образование, в котором возможно использование блокчейна для защиты интеллектуальной собственности при создании обучающих алгоритмов.
- Финансы, в которых возможна защита алгоритмов для анализа рыночных данных и автоматизация обработки транзакций.

К числу ограничений технологии блокчейна относятся:

- Высокие затраты на вычисления и хранение, особенно при обработке больших данных.
- Ограничения по скорости транзакций в распределённых системах [19].
- Юридические аспекты при обработке персональных данных [20].

Отметим, что задача оценивания блокчейн-систем с помощью тестирования, верификации и валидации ставилась в [21] и была развита в [22].

Современные криптографические методы защиты предоставляют мощные инструменты для обеспечения конфиденциальности данных и целостности моделей. Несмотря на текущие ограничения, такие как высокая вычислительная сложность и сложности интеграции, их использование необходимо для минимизации рисков утечек данных и атак на модели. Исследования в этой области должны быть направлены на улучшение производительности и разработку гибридных решений. Для этого, в том числе, должны быть проведены глубокие математические исследования (к примеру, как в [23-25]) для исследования различных характеристик [26-28].

Применение криптографических методов в системах машинного обучения

Современные криптографические методы играет ключевую роль в обеспечении защиты данных и моделей МО. Они не только минимизируют риски, но и способствуют созданию доверительных отношений между пользователями и технологиями.

Рассмотрим сначала облачные сервисы. Гомоморфное шифрование позволяет использовать облачные вычисления для обработки данных без риска их раскрытия. Например, компании могут безопасно анализировать зашифрованные данные клиентов, обеспечивая защиту конфиденциальной информации в финансовых транзакциях или медицинских исследованиях. Это особенно актуально в условиях использования облачных платформ, где данные пользователей передаются через общедоступные сети. Примером может служить безопасный анализ геномных данных, где гомоморфное шифрование обеспечивает полную конфиденциальность даже при выполнении сложных вычислений на стороне облачного провайдера [3, 4, 13].

Также облачные сервисы используют многосторонние вычисления для совместного анализа данных из различных источников. Например, несколько медицинских учреждений могут использовать облачную платформу для обучения общей модели на локальных данных, не передавая информацию на центральный сервер. Это гарантирует как сохранение конфиденциальности, так и выполнение сложных вычислений без риска компрометации данных [3].

Федеративное обучение [30] сочетает многосторонние вычисления и дифференциальную приватность, обеспечивая безопасное обучение моделей на распределённых данных. В рамках этой технологии устройства обучают модели локально и передают только обновления, защищённые с помощью криптографических методов. Например, в мобильных приложениях федеративное обучение применяется для разработки рекомендательных систем, анализирующих поведение пользователей, не раскрывая их личных данных.

Дифференциальная приватность [12] в федеративном обучении добавляет шум к передаваемым обновлениям, защищая от атак на восстановление данных. Это позволяет использовать модели в критически важных областях, таких как здравоохранение, где данные о пациентах не могут быть раскрыты. Примером может быть использование федеративного обучения для создания алгоритмов диагностики заболеваний, где клиники объединяют усилия, не раскрывая данные пациентов.

Федеративное обучение также используется в индустриальных приложениях, таких как IoT-системы, где устройства собирают данные с сенсоров. Многосторонние вычисления позволяют агрегировать данные с тысяч устройств, сохраняя их конфиденциальность. Это особенно важно в случаях, когда устройства расположены в разных странах, и данные подчиняются различным законам о защите информации [30].

Блокчейн-технология используется для записи изменений моделей и обеспечения их подлинности. Это позволяет защитить интеллектуальную собственность разработчиков и гарантировать прозрачность всех операций, связанных с данными и моделями. Например, компании могут фиксировать в блокчейне каждую итерацию обучения модели, обеспечивая невозможность подделки или изменения её параметров [18, 15, 19].

Блокчейн также применяется для управления доступом к данным и моделям в распределённых системах. Например, смарт-контракты могут автоматически предоставлять доступ к модели только авторизованным пользователям, что исключает несанкционированное использование. Это особенно важно в финансовой сфере, где нарушение конфиденциальности может привести к значительным потерям [18, 15, 19].

Современные системы машинного обучения всё чаще используют комбинированные подходы для повышения уровня безопасности. Например, объединение блокчейна, гомоморфного шифрования и дифференциальной приватности позволяет создавать распределённые системы, которые обеспечивают как конфиденциальность данных, так и прозрачность операций. Такие решения находят применение в разработке платформ для медицинских исследований, где требуется совместная обработка данных большого числа участников [31].

Комбинированные методы также используются в сфере искусственного интеллекта для обучения моделей с открытым доступом. Например, использование блокчейна для контроля версии модели и гомоморфного шифрования для защиты входных данных гарантирует безопасность и прозрачность разработки. Эти подходы позволяют объединить преимущества разных технологий, минимизируя их ограничения [19, 32].

Сравнительный анализ предложенных методов защиты

Современные криптографические методы обладают разным уровнем защиты и подходят для решения различных задач, связанных с обеспечением безопасности систем МО. Оценим эффективность предложенных выше подходов.

Гомоморфное шифрование (ГШ) обеспечивает высокий уровень конфиденциальности, позволяя производить вычисления над зашифрованными данными. Этот метод эффективен для задач, требующих обработки данных в облачных вычислениях, таких как обучение моделей на конфиденциальных данных. Однако высокая вычислительная сложность остаётся значительным ограничением [3, 4].

Многосторонние вычисления (MPC) подходят для обучения моделей на распределённых данных, где данные находятся у нескольких участников. Они обеспечивают конфиденциальность данных, однако требуют больших ресурсов для реализации и координации между участниками [4, 10].

Дифференциальная приватность (DP) наиболее применима в задачах, связанных с анализом данных. Она обеспечивает защиту от восстановления исходных данных, добавляя случайный шум. Однако избыточный шум может снижать точность предсказаний модели [11, 12].

Блокчейн демонстрирует высокую устойчивость к модификации данных, что особенно полезно для отслеживания истории изменений модели или управления доступом. Однако недостатками являются высокая вычислительная стоимость и ограниченная скорость обработки транзакций [14, 29, 17, 18].

Таким образом, выбор метода зависит от задач системы МО и требований к безопасности данных. Например, для финансовых данных предпочтительнее использовать MPC или гомоморфное шифрование, а в системах с большим числом участников – дифференциальную приватность.

Каждый криптографический метод имеет ограничения, связанные с:

- Вычислительными затратами: гомоморфное шифрование требует значительных ресурсов, особенно в задачах реального времени, таких как онлайн-обработка данных или обучение глубоких нейронных сетей [3, 10].
- Масштабируемостью: методы MPC могут быть сложны для реализации в системах с большим числом участников, так как требуют значительного взаимодействия между сторонами [4, 11].
- Интеграцией в существующие системы: интеграция криптографических методов требует изменения архитектуры моделей и инфраструктуры, что может быть трудоёмким процессом [11, 17].
- Точностью: добавление шума в рамках дифференциальной приватности может влиять на производительность модели, особенно в задачах с высокими требованиями к точности [3, 12].

К числу перспектив и направлений развития предложенных методов можно отнести

- Оптимизацию методов гомоморфного шифрования, под чем мы понимаем то, что исследования сосредоточены на улучшении производительности схем, таких как CKKS, для повышения скорости операций над зашифрованными данными [4, 13].
- Использование комбинированных подходов, под чем мы понимаем то, что происходит интеграция различных методов, таких как блокчейн и MPC, для усиления безопасности и улучшения масштабируемости. Например, использование блокчейна для управления доступом к моделям, а MPC – для совместного обучения на данных [29, 15].
- Автоматизация настройки параметров, под чем мы понимаем то, что происходит разработка инструментов для автоматической настройки параметров, таких как ε в дифференциальной приватности, чтобы минимизировать влияние на точность моделей [12, 19].
- Разработка специализированного оборудования, под чем мы понимаем то, что происходит использование аппаратных ускорителей для выполнения криптографических операций, таких как гомоморфное шифрование, что позволит снизить накладные расходы [10, 32].

Сравнительный анализ показывает, что ни один из методов не может считаться универсальным решением для всех задач безопасности МО. Наиболее перспективным направлением является комбинирование методов, которое позволяет компенсировать недостатки одного подхода за счёт преимуществ другого. В частности, блокчейн может обеспечивать неизменность данных, MPC – конфиденциальность, а дифференциальная приватность – защиту от восстановления исходных данных. Исследования в области интеграции этих методов, а также разработка решений, обеспечивающих их масштабируемость и эффективность, являются ключевыми для дальнейшего развития системы защиты МО.

Заключение

В данной работе рассмотрены основные угрозы, связанные с безопасностью систем машинного обучения (МО), и современные криптографические методы их предотвращения, включая гомоморфное шифрование, многосторонние вычисления, дифференциальную приватность и блокчейн. Проведённый анализ продемонстрировал эффективность этих подходов в защите данных и моделей от утечек, атак и несанкционированного доступа. Несмотря на значительный прогресс в данной области, остаются ограничения, такие как высокая вычислительная сложность, масштабируемость и интеграция в существующие системы.

Перспективным направлением является комбинирование различных методов для повышения устойчивости систем МО, что открывает новые возможности для их применения в критически важных областях, таких как здравоохранение, финансы и облачные вычисления.

Литература

1. *Shokri R., Stronati M., Song C., Shmatikov V*. Membership Inference Attacks Against Machine Learning Models // 2017 IEEE Symposium on Security and Privacy (SP), San Jose, CA, USA, 2017, pp. 3-18, doi: 10.1109/SP.2017.41.
2. *Shafahi A., Najibi M., Ghiasi A., Xu Z., Dickerson J., Studer C., Davis L.S., Taylor G., Goldstein T.* Adversarial training for free // Proceedings of the 33rd International Conference on Neural Information Processing Systems, 2019, pp. 3358-3369.
3. *Gentry C*. Fully homomorphic encryption using ideal lattices // Proceedings of the 41st Annual ACM Symposium on Theory of Computing. STOC 2009, 2009, pp. 169-178.
4. *Чеон Дж. Х., Ким А., Ким М., Сонг Ю*. Гомоморфное шифрование для арифметики приближённых чисел / Дж. Х. Чеон, А. Ким, М. Ким, Ю. Сонг. // Труды ASIACRYPT. 2017.
5. *Маршалко Г.Б*. Национальная и международная стандартизация российских криптографических алгоритмов. PKI forum 2014, Санкт-Петербург, Russia, 16-18 сентября 2014 // PKI forum: [сайт]. https://pki-forum.ru/files/files/archive_2014/11%20marshalko.pdf (дата обращения: 26.01.2025).
6. *Панков К.Н., Миронов Ю.Б*. Использование постквантовых алгоритмов в задачах защиты информации в телекоммуникационных системах. М.: Горячая линия – Телеком, 2023. 236 с. ISBN 978-5-9912-1015-7. EDN MTJUJL
7. Распоряжение Правительства РФ от 11 июля 2023 г. № 1856-р. Об утверждении Концепции регулирования отрасли квантовых коммуникаций в РФ до 2030 г. https://www.garant.ru/products/ipo/prime/doc/407297268/ (дата обращения: 26.01.2025).
8. *Bogdanov D., Laur S., Willemson J.* Sharemind: A Framework for Fast Privacy-Preserving Computations // Computer Security – ESORICS 2008. ESORICS 2008. Lecture Notes in Computer Science, vol 5283. Springer, Berlin, Heidelberg, 2008, pp. 192-206 https://doi.org/10.1007/978-3-540-88313-5_13
9. *Aono Y., Hayashi T., Phong L.T., Wang L*. Privacy-preserving logistic regression with distributed data sources via homomorphic encryption // IEICE TRANSACTIONS on Information and Systems. 2016 Aug 1, no. 99(8), pp. 2079-2089.
10. *Запечников С.В*. Криптографическая защита процессов обработки информации в недоверенной среде: достижения, проблемы, перспективы // Вестник современных цифровых технологий. 2019. № 1. С. 4-16. EDN JDKVZQ
11. *Dwork C., Roth A*. The Algorithmic Foundations of Differential Privacy, now, 2014, 228 p. doi: 10.1561/0400000042.
12. *Abadi M., Chu A., Goodfellow I.J., McMahan H.B., Mironov I., Talwar K., Zhang L*. Deep Learning with Differential Privacy // Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, 2016, pp. 308-318, https://doi.org/10.1145/2976749.2978318
13. *Панков К.Н.* Использование криптографических средств для сквозных цифровых технологий на примере систем распределенного реестра // Технологии информационного общества : Материалы XII Международной отраслевой научно-технической конференции, Москва, 14-15 марта 2018 года. Том 1. М.: Издательский дом Медиа Паблишер, 2018. С. 365-366. EDN UUHHSM
14. *Nakamoto S*. Bitcoin: A Peer-to-Peer Electronic Cash System, 31.10.2008. // Satoshi Nakamoto Institute: [сайт]. https://cdn.nakamotoinstitute.org/docs/bitcoin.pdf (дата обращения: 26.01.2025).
15. *Маршалко Г.Б*. Блокчейн — трезвый взгляд, реальные применения и текущие проблемы. 2019. // CTCrypt: [сайт]. https://ctcrypt.ru/files/files/2019/materials/30_Marshalko.pdf (дата обращения: 26.01.2025).
16. *Панков К.Н., Миронов Ю.Б*. Применение квантовых методов в задачах защиты информации. М.: Горячая линия – Телеком, 2022. 212 с. ISBN 978-5-9912-1014-0. EDN MDQZBY
17. Белая книга цифровой экономики, 2023. // Белая книга цифровой экономики: онлайн-версия: [сайт]. https://цифроваяэкономика.рф/upload/uf/833/8rnf976ag03i3to5el2o69cy7hrk3e17/White_paper_2023_.pdf (дата обращения: 26.01.2025).
18. *Маршалко Г.Б.* Уязвимости блокчейн и других новых технологий. 2017. // Digital Russia: [сайт]. https://цифроваяэкономика.рф/upload/uf/833/https://d-russia.ru/wp-content/uploads/2017/09/marshalko.pdf (дата обращения: 26.01.2025).
19. *Маршалко Г.Б.* Проблемы технологии блокчейн с точки зрения стандартизации – 2017. // PKI-Форум: [сайт]. https://pki-forum.ru/files/files/2017/29_Marshalko.pdf (дата обращения: 26.01.2025).
20. *Pankov K*. Enumeration of Boolean Mapping with Given Cryptographic Properties for Personal Data Protection in Blockchain Data Storage // Conference of Open Innovations Association, FRUCT. 2019. No. 24, pp. 300-306. DOI 10.23919/FRUCT.2019.8711894. EDN BOVLMR

21. *Pankov K.N*. Testing, Verification and Validation of Distributed Ledger Systems // 2020 Systems of Signals Generating and Processing in the Field of on Board Communications, Moscow, 19-20 марта 2020 г. Moscow: Institute of Electrical and Electronics Engineers Inc., 2020. P. 9078541. DOI 10.1109/IEEECONF48371.2020.9078541. EDN CITRFX

22. *Панков К.Н., Эйнман А.Д.* Сертификация систем распределенного реестра как инструмент обеспечения информационной безопасности // REDS: Телекоммуникационные устройства и системы. 2021. Т. 11, № 2. С. 37-49. EDN CGQQYR

23. *Панков К.Н*. Локальная предельная теорема для распределения части вектора весов подфункций компонент случайного двоичного отображения // Математические вопросы криптографии. 2014. Т. 5, № 3. С. 49-80. EDN TFNXVD

24. *Pankov K.N*. Improved asymptotic estimates for the numbers of correlation-immune and k-resilient vectorial Boolean functions // Discrete Mathematics and Applications. 2019. Vol. 29, No. 3, pp. 195-213. DOI 10.1515/dma-2019-0018. EDN CFOLBU

25. *Kamlovskii O.V., Pankov K.N*. Some Classes of Balanced Functions over Finite Fields with a Small Value of the Linear Characteristic // Problems of Information Transmission. 2022. Vol. 58, No. 4, pp. 389-402. DOI 10.1134/s0032946022040093. EDN QSFFJO

26. *Панков К.Н*. Оценки скорости сходимости в предельных теоремах для совместных распределений части характеристик случайных двоичных отображений // Прикладная дискретная математика. 2012. № 4(18). С. 14-30. EDN PJPPCX

27. *Панков К.Н*. Оценки мощности классов отображений, применяемых в протоколах квантового распределения ключей // Наукоемкие технологии в космических исследованиях Земли. 2022. Т. 14, № 4. С. 4-18. DOI 10.36724/2409-5419-2022-14-4-4-18. EDN QKXSQK

28. *Панков К.Н*. Оценки мощности классов отображений, применяемых в протоколах квантового распределения ключей // Наукоемкие технологии в космических исследованиях Земли. 2022. Т. 14, № 4. С. 4-18. DOI 10.36724/2409-5419-2022-14-4-4-18. EDN QKXSQK

29. *Чжэн З., Се С., Дай Х., Чен Х., Ван Х.* Обзор технологии блокчейн: архитектура, консенсус и будущие тенденции // Международный конгресс IEEE по большим данным. 2017.

30. *Zheng C., Wang L., Xu Z., Li H*. Optimizing Privacy in Federated Learning with MPC and Differential Privacy // 2024 3rd Asia Conference on Algorithms, Computing and Machine Learning (CACML 2024), March 22-24, 2024, Shanghai, China. ACM, New York, NY, USA 5 Pages. https://doi.org/10.1145/3654823.3654854

31. *Almalawi A., Hassan S., Fahad A., Khan A.I.* A Hybrid Cryptographic Mechanism for Secure Data Transmission in Edge AI Networks. // Int J Comput Intell Syst 17, 24 (2024). https://doi.org/10.1007/s44196-024-00417-8

32. *Маршалко Г.Б., Дыгин Д.М., Лавриков И.В*. О криптографии и валютах в криптовалютах – 2016. // Конференция «РусКрипто»: [сайт]. https://ruscrypto.ru/resource/archive/rc2016/files/07_dygin_lavrikov_marshalko.pdf (дата обращения: 26.01.2025).

ИМИТАЦИОННАЯ МОДЕЛЬ КРИПТОГРАФИЧЕСКОГО ПРОТОКОЛА ШИФРОВАНИЯ

Михалевич Игорь Феодосьевич
Российский университет транспорта, доцент, к.т.н., старший научный сотрудник, Москва, Россия
mif-orel@mail.ru

Жеребятин Илья Андреевич
Российский университет транспорта, студент, Москва, Россия
opoxil@mail.ru

Аннотация
В данной работе представлена имитационная модель протокола конфиденциального обмена, основанного на моноалфавитном шифре замены, шифре перестановки и алгоритме Диффи-Хеллмана, с использованием языка программирования Python. Особое внимание уделяется тому, что для глубокого понимания функционирования этих криптографических систем и их корректного применения необходимы знания в области математических наук, а также умение интерпретировать полученные результаты. Реализация ИМ протокола осуществляется в виде приложения, что позволяет продемонстрировать принципы работы различных шифров и алгоритмов.

Ключевые слова
Алгоритм Диффи-Хеллмана, безопасность информации, визуализация вычислений, имитационная модель, конфиденциальный обмен, криптография, кроссплатформенное приложение, математические науки, моноалфавитный шифр замены, шифр перестановки, Python

Введение

В современном мире обеспечение безопасности информации становится все более актуальным. Одной из ключевых проблем данной сферы является нехватка квалифицированных кадров в области информационной безопасности. Это во многом связано с трудностями в подготовке специалистов, владеющих цифровыми технологиями разработки сложных математических концепций, особенно в криптографии. Традиционный подход к обучению, ориентированный лишь на теоретическое изложение материала, может создавать состояние стресса, вызывать у обучаемых чувство неуверенности, усугубляя реальные трудности и снижая интерес к предмету. Чтобы преодолеть эти барьеры, необходимо использовать методы визуализации, которые позволяют сделать сложные идеи более доступными и понятными, создавая условия успешного развития современной цифровой образовательной среды [1].

В данной работе разработана имитационная модель (ИМ) протокола конфиденциального обмена, основанного на моноалфавитном шифре замены, шифре перестановки и алгоритме Диффи-Хеллмана, выполненная на языке Python. Предложенная ИМ позволяет в дружелюбной форме контролировать полноту теоретических знаний и корректность их практического применения, предоставляет интерактивную среду изучения основ криптографии. Использование ИМ в обучении криптографии может значительно повысить интерес студентов, а также улучшить их навыки в области информационной безопасности, что является важным шагом к подготовке квалифицированных кадров в данной сфере.

Выбор состава имитационной модели

Приоритетность выбора состава ИМ протокола конфиденциального обмена основывается на нескольких ключевых факторах, которые делают данную тему особенно значимой в контексте современных требований к информационной безопасности [2, 3].

Во-первых, моноалфавитный шифр замены и шифр перестановки представляют собой классические методы шифрования, которые служат основой для понимания более сложных криптографических систем. Их простота и наглядность позволяют эффективно иллюстрировать основные принципы криптографии. Использование этих шифров, реализованных форме ИМ, позволяет легко демонстрировать процессы шифрования и расшифрования, что способствует лучшему усвоению материала.

Во-вторых, алгоритм Диффи-Хеллмана, как один из базовых протоколов выработки ключей, играет важную роль в обеспечении безопасности коммуникаций. Он позволяет участникам создавать надежные ключи, которые потом могут использоваться для связи по незащищённым каналам, что является критически важным для современных систем передачи данных. Включение этого алгоритма в ИМ протокола конфиденциального обмена позволяет продемонстрировать, как классические шифры могут эффективно сочетаться с современными методами криптографии для достижения высокого уровня безопасности [4, 5].

Наконец, реализация ИМ протокола на языке Python предоставляет возможность создания интерактивного приложения, что делает процесс обучения более увлекательным и доступным. Python, благодаря своей простоте и широким возможностям для работы с данными, идеально подходит для разработки ИМ криптографических систем. Это позволяет не только визуализировать теоретические аспекты, но и проводить эксперименты с различными параметрами шифрования, углубляя понимание процессов, лежащих в основе криптографии.

Таким образом, выбор состава ИМ обоснован как с точки зрения образовательных целей, так и с точки зрения практического применения в области информационной безопасности, что делает разработку ИМ протокола конфиденциального обмена актуальным и полезным проектом.

Описание моделируемой системы

Моделируемая система представляет собой протокол конфиденциального обмена, который интегрирует три ключевых компонента: моноалфавитный шифр замены, шифр перестановки и алгоритм Диффи-Хеллмана. Каждый из этих компонентов играет важную роль в обеспечении безопасности и конфиденциальности передаваемой информации.

Принцип работы моноалфавитного шифра замены показан на рисунке 1.

Позиция символа в алфавите шифра (мощность n = 80)

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39
Шифровальная таблица при ключе равном 9																																							
А	Б	В	Г	Д	Е	Ё	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	а	б	в	г	д	е	ё
К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	а	б	в	г	д	е	ё	ж	з	и	й	к	л	м	н	о	п	р

Позиция символа в алфавите шифра (мощность n = 80)

40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79
Шифровальная таблица при ключе равном 9																																							
ж	з	и	й	к	л	м	н	о	п	р	с	т	у	ф	х	ц	ч	ш	щ	ъ	ы	ь	э	ю	я	-	.	/	0	1	2	3	4	5	6	7	8	9	
с	т	у	ф	х	ц	ч	ш	щ	ъ	ы	ь	э	ю	я	-	.	/	0	1	2	3	4	5	6	7	8	9		А	Б	В	Г	Д	Е	Ё	Ж	З	И	Й

Исходное сообщение

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39
1	У	в	а	ж	а	е	м	ы	й		И	г	о	р	ь		Ф	е	д	о	с	ь	е	в	и	ч	.		Р	а	з	р	а	б	о	т	а	н	о	
2	с	е	м	е	й	с	т	в	о		п	р	о	т	о	к	о	л	о	в		к	о	н	ф	и	д	е	н	ц	и	а	л	ь	н	о	г	о		о
3	б	м	е	н	а		н	а		о	с	н	о	в	е		м	о	н	о	а	л	ф	а	в	и	т	н	о	г	о		ш	и	ф	р	а		с	
4	и	с	п	о	л	ь	з	о	в	а	н	и	е	м		м	е	х	а	н	и	з	м	о	в		з	а	м	е	н	ы	,		п	е	р	е	с	т
5	а	н	о	в	о	к		и		а	л	г	о	р	и	т	м	а		Д	и	ф	ф	и	-	Х	е	л	м	а	н	а	.		Д	л	я		т	е
6	с	т	а		ш	и	ф	р	у	ю		п	а	р	о	л	ь		д	л	я		а	р	х	и	в	а	.		Э	т	о		а	б	в	г	д	е
7	ё	ж																																						

Зашифрованное сообщение

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39
1	Ю	м	к	С	к	п	Ч	ч	Ф	Й	У	н	Щ	Ы	ч	Й	Я	п	о	Щ	Ь	ч	п	м	У	в	9	Й	Ы	к	Т	Ы	к	л	Щ	Э	к	Ш	Щ	Й
2	Ь	п	Ч	п	Ф	Ь	Э	м	Щ	Й	Ъ	Ы	Щ	Э	Щ	Х	Щ	Ц	Щ	м	Й	Х	Щ	Ш	Я	У	о	п	Ш	б	У	к	Ц	ч	Ш	Щ	н	Щ	Й	Щ
3	л	Ч	п	Ш	к	Й	Ш	к	Й	Щ	Ь	Ш	Щ	м	п	Й	Ч	Щ	Ш	Щ	к	Ц	Я	к	м	У	Э	Ш	Щ	н	Щ	Й	г	У	Я	Ы	к	Й	Ь	Й
4	У	Ь	Ъ	Щ	Ц	ч	Т	Щ	м	к	Ш	У	п	Ч	Й	Ч	п	а	к	Ш	У	Т	Ч	Щ	м	Й	Т	к	Ч	п	Ш	ч	##	Й	Ъ	п	Ы	п	Ь	Э
5	к	Ш	Щ	м	Щ	Х	Й	У	Й	к	Ц	н	Щ	Ы	У	Э	Ч	к	Й	о	У	Я	Я	У	8	а	п	Ц	Ч	к	Ш	к	9	Й	о	Ц	ч	Й	Э	п
6	Ь	Э	к	Й	г	У	Я	Ы	Ю	ч	Й	Ъ	к	Ы	Щ	Ц	ч	Й	о	Ц	ч	Й	к	Ы	а	У	м	к	9	Й	ч	Э	Щ	Й	к	л	м	н	о	п
7	Р	С	Й	Й	Й	Й	Й	Й	Й	Й	Й	Й	Й	Й	Й	Й	Й	Й	Й	Й	Й	Й	Й	Й	Й	Й	Й	Й	Й	Й	Й	Й	Й	Й	Й	Й	Й	Й	Й	Й

Рис. 1. Принцип работы моноалфавитного шифра замены

Этот метод шифрования основывается на замене каждого символа исходного сообщения на другой символ из фиксированного алфавита.

Алфавит – это конечный набор символов, используемых для представления информации в языке или системе. Алфавит может состоять из букв, цифр, знаков препинания и других символов, которые могут комбинироваться для формирования слов, предложений и других структур. Так, например, русский алфавит содержит 33 буквы английский - 26 букв, а вот таблицы ASCII включают 256 кодов букв нескольких языков, цифр, знаков препинания и иных символов.

Общее количество уникальных символов, входящих в алфавит, определяет его мощность. На рисунке 1 алфавит содержит прописные и строчные русские буквы, цифры и другие уникальные символы, общим числом 80. Мощность данного алфавита равна 80. Созданный на его основе шифр прост в реализации и позволяет наглядно продемонстрировать основные принципы шифрования.

В рамках ИМ пользователи могут генерировать свои собственные ключи для замены, что делает шифрование индивидуальным. Этот компонент обеспечит базовую защиту данных и станет первым уровнем шифрования. Однако, несмотря на простоту и наглядность, моноалфавитный шифр замены имеет определенные недостатки. Одним из наиболее значительных является уязвимость к криптоанализу. Поскольку каждый символ заменяется на фиксированный, частота появления символов в зашифрованном тексте будет отражать частоту в открытом тексте. Это открывает возможности для криптоаналитиков, которые могут использовать частотность для восстановления исходного сообщения, особенно если текст достаточно длинный. Следовательно, хотя моноалфавитный шифр замены может служить хорошей основой для понимания криптографии и обеспечения базовой защиты данных, его недостатки делают его неэффективным для защиты конфиденциальной информации в современных условиях. Тем не менее этот метод остаётся полезным в образовательных целях для тех, кто только начинает изучать принципы шифрования и защиты информации.

На рисунке 2 показан принцип работы шифра перестановки. Данный метод шифрования изменяет порядок символов в сообщении, но не меняет сами символы, что необходимо для создания дополнительного уровня сложности шифрованного сообщения, делая его менее уязвимым для атак, основанных на частотном анализе.

Номер шага																																				
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39
1	У	в	а	ж	а	е	м	ы	й		И	г	о	р	ь		Ф	е	д	о	с	ь	е	в	и	ч	.		Р	а	з	р	а	б	о	т	а	н	о
2		с	е	м	е	й	с	т	в	о		п	р	о	т	о	к	о	л	о	в		к	о	н	ф	и	д	е	н	ц	и	а	л	ь	н	о	г	о
3		о	б	м	е	н	а		н	а		о	с	н	о	в	е		м	о	н	о	а	л	ф	а	в	и	т	н	о	г	о		ш	и	ф	р	а
4		с		и	с	п	о	л	ь	з	о	в	а	н	и	е	м		м	е	х	а	н	и	з	м	о	в		з	а	м	е	н	ы	,		п	е	
5	р	е	с	т	а	н	о	в	о	к		и		а	л	г	о	р	и	т	м	а		Д	и	ф	ф	и	-	Х	е	л	м	а	н	а	.		Д	
6	л	я		т	е	с	т	а		ш	и	ф	р	у	ю		п	а	р	о	л	ь		д	л	я		а	р	х	и	в	а	.		Э	т	о		
7	а	б	в	г	д	е	ё	ж																															

Номер шага																																							
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39
1	У		д		а	с	о	н	л	м	с	о	о		з	м	в		о	г	и	а	т	р	ь	и	а													
2	в	И	о	Р	н	т	к	ф	ь	е	н	а	г	с	о	е		п	в	о	ф	н	е	у		в	б												
3	а	г	с	а	о	в	о	и	н	н	о	л	о		в	х	з	е	о	р	ф	а	с	ю	д	а	в												
4	ж	о	ь	з		о	л	д	о	а	в	ф		и	а	а	а	р	к	и	и	.	т		л	.	г												
5	а	р	е	р	с		о	е	г		е	а	ш	с	н	н	м	е		т	-		а	п	я		д												
6	е	ь	в	а	е	п	в	н	о	н		в	и	п	и	и	е	с	и	м	Х	Д		а		Э	е												
7	м		и	б	м	р		ц		а	м	и	ф	о	е	з	н	т		а	е	л	ш	р	а	т	ё												
8	ы	Ф	ч	о	е	о	к	и	о		о	т	р	л	м	м	ы	а	а		л	я	и	о	р	о	ж												
9	й	е	.	т	й	т	о	а	б	о	н	н	а	ь		о	,	н	л	Д	м		ф	л	х														

Рис. 2. Принцип работы шифра перестановки

В ИМ реализуется возможность выбора различных схем перестановки, что позволяет пользователям адаптировать шифрование под свои нужды и повышать уровень безопасности. Схема перестановки может быть различной. Пользователи могут выбирать, например, фиксированный порядок, при котором каждый символ будет перемещаться на определённое количество позиций, или более сложные схемы, такие как перемешивание символов в группах. Это обеспечивает создание уникального шифра для каждого сообщения, что значительно усложняет задачу криптоанализа, так как, даже если частота появления символов будет известна, их порядок будет оставаться скрытым. Пользователи могут экспериментировать с различными ключами и схемами, создавая уникальные шифры для каждого сообщения, что в свою очередь повышает уровень защиты передаваемой информации.

Принцип работы алгоритма Диффи-Хеллмана показан на рисунке 3. Алгоритм основывается на сложности дискретного логарифмирования, позволяет двум сторонам вырабатывать секретные ключи путем обмена нужными для этого параметрами по открытым каналам связи и обеспечивает защиту от взлома ключей.

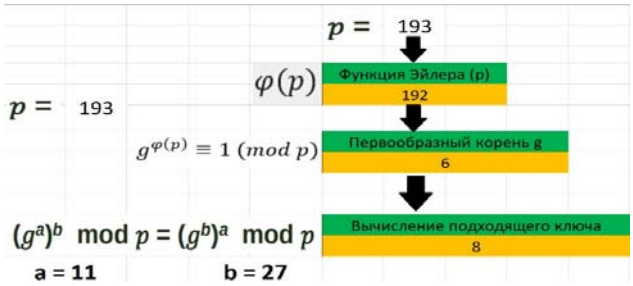


Рис. 3. Принцип работы алгоритма Диффи-Хеллмана

В рамках ИМ пользователи могут обмениваться открытыми ключами и сохранять в тайне закрытые ключи, что позволяет им осуществлять шифрование и расшифрование сообщений с использованием комбинации моноалфавитного шифра и шифра перестановки.

Описание имитационной модели

Имитационная модель протокола конфиденциального обмена сообщениями реализована в виде кроссплатформенного приложения. Такой подход облегчает понимание сложных криптографических алгоритмов и предоставляет пользователям возможность экспериментировать с различными методами шифрования, включая моноалфавитный шифр замены и шифр перестановки.

Для разработки имитационной модели с расширенной функциональностью визуализации вычислительных процессов использовалась среда разработки IDE и язык программирования Python. В процессе разработки были созданы классы и функции для реализации различных компонентов шифрования [6, 7].

Функционирование имитационной модели

При запуске программы необходимо ввести значение простого числа *p*, для которого будут генерироваться промежуточные открытый и закрытый ключи на каждой стороне, на основе которых будет выработан общий для сторон ключ симметричного шифрования. На рисунке 4 показано окно вывода текста, в котором указаны параметры *a, b* которые позволяют вычислить ключ для дальнейшего шифрования.

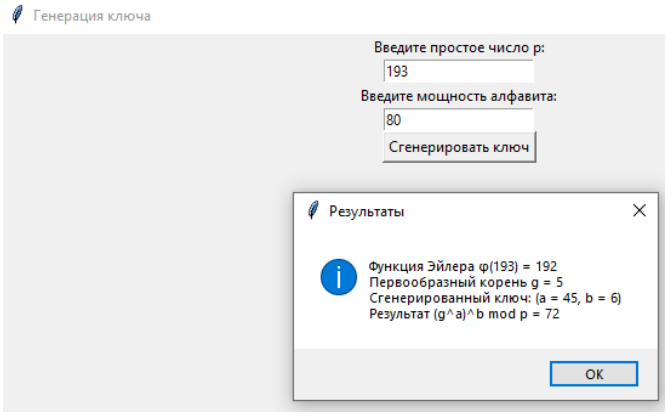


Рис. 4. Окно ввода исходных данных

На данном шаге ИМ:

– вычисляет наибольший общий делитель двух чисел *a* и *b* с использованием алгоритма Евклида;

– находит все натуральные числа меньшие или равные *p*, которые являются взаимно простыми с *p* с помощью функции Эйлера и первообразного корня;

– находит числа, которые могут быть корнями в группе чисел по модулю *p*;

– производит возведение в степень с учетом остатка от деления.

Графический интерфейс ИМ включает в себя:

– главное окно приложения с заголовком «Модульная арифметика»;

– поле для ввода натурального числа *p*;

– кнопку, которая запускает процесс вычисления;

– текстовое поле для отображения результатов вычислений, с возможностью прокрутки, если вывод превышает видимую область.

Алгоритм работы ИМ:

– запуск работы программы посредством кнопки «Сгенерировать ключ»;

– считывание значения *p* из текстового поля;

– проверка корректность ввода;

– выполнение необходимых вычислений, включая нахождение взаимно простых чисел, генерацию случайных чисел;

– вывод результатов в текстовое поле;

– обработка возможных ошибок ввода, с отображением сообщений об ошибках с помощью диалогового окна.

После выполнения вышеперечисленных операций программа выведет на экран зашифрованное сообщение, которое можно передавать далее по открытым каналам (рис. 5).

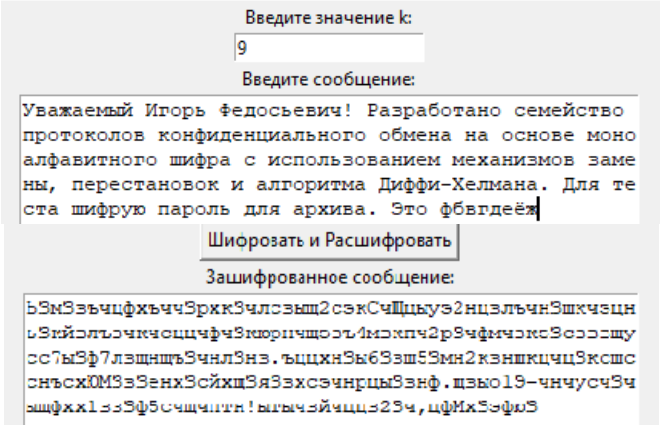


Рис. 5. Результат шифрования

Моделирование техник взлома протокола

Для успешного предотвращения компьютерных преступлений необходим постоянный контроль сетей на предмет наличия уязвимостей, то есть системный мониторинг безопасности, в том числе путем моделирования атак злоумышленников.

В данной работе будет проанализирован ряд вариантов атак, таких как:

– прямая подстановка;

– нахождение закономерностей.

Анализ методов несанкционированного дешифрования необходим с целью их предотвращения.

Первым вариантом взлома предложенных методов шифрования будет автоматический подбор с применением brute-force. Анализируя зашифрованное сообщение, злоумышленник приходит к выводу, что алфавит состоит из заглавных и прописных букв, цифр и знаков пунктуации. Следовательно, можно допустить, что мощность алфавита протокола равна 80 символам. Кроме того, злоумышленник предполагает, что для обоих методов шифрования использовался один и тот же ключ. Из этого следует: максимальное количество вариантов зашифрованного сообщения равняется 80, что позволяет просто с помощью короткого цикла перебрать все возможные варианты и найти ключ.

Метод дешифрования, основанный на прямом подборе ключа по мощности алфавита, представляет собой прямолинейный, но эффективный подход к взлому сообщений, зашифрованных с использованием моноалфавитного шифра замены и шифра перестановки. Данный метод основывается на идее о том, что, зная мощность алфавита, можно генерировать все возможные ключи и проверять их на соответствие дешифрованному тексту.

Это позволяет использовать все возможные комбинации символов для восстановления исходного сообщения. При использовании моноалфавитного шифра замены, где каждый символ заменяется на другой, прямой подбор ключа может быть особенно эффективным, поскольку количество возможных замен ограничено мощностью алфавита. Например, если используется алфавит из 80 символов, общее количество возможных комбинаций для замены может достигать значительных величин, однако, благодаря структурным особенностям языка, таких как частота букв и типичные слова, можно кардинально сократить количество проверок.

Когда речь идет о шифре перестановки, ситуация несколько усложняется, так как в данном случае необходимо учитывать порядок символов. Однако, если известна мощность алфавита и количество символов в сообщении, мы можем генерировать все возможные перестановки и проверять их на соответствие. Несмотря на то, что количество возможных перестановок растет факториально с увеличением длины сообщения, данный метод может быть полезен в случае коротких текстов, где время на проверку всех комбинаций становится приемлемым. В итоге, как видно из рисунка 6 при j=9 (на 9-й итерации) был подобран правильный ключ, что наглядно демонстрирует значительную уязвимость данного шифра к атаке brute-force.

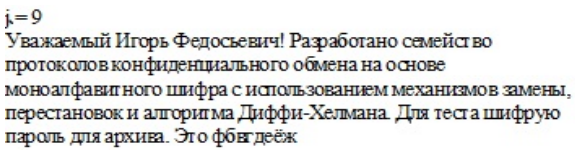


Рис. 6. Взлом протокола методом «грубой силы» при одинаковых ключах на шагах протокола

Теперь рассмотрим другой вариант зашифрованного сообщения, представленный на рисунке 7. Оно получено путем применения различных ключей на шагах протокола (разные ключи для каждого метода шифрования).

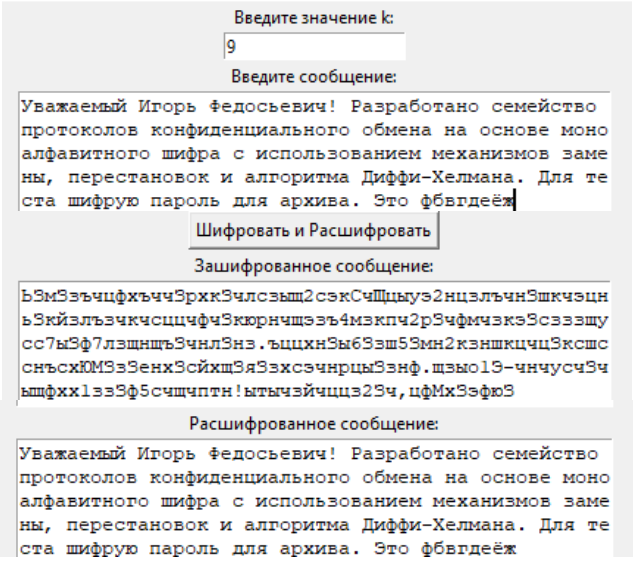


Рис. 7. Результаты шифрования и расшифрования при различных ключах на шагах протокола

Для дешифровки сообщения, зашифрованного с применением различных ключей, также злоумышленник может использовать взлом с помощью brute-force. Однако в данном случае потребуется больше времени на подбор нужного варианта, что делает перебор более затруднительным, но возможным. В нашем случае, взломщику потребовалось всего две минуты, чтобы взломать ключи (рис. 8, ключи 9 и 41) и найти правильный вариант начального текста.

k для моноалфавитного шифра замены = 9
k для шифра перестановки = 41
Уважаемый Игорь Федосьевич! Разработано семей-ство протоколов конфиденциального обмена на основе моноалфавитного шифра с использованием механизмов замены, перестановок и алгоритма Диффи-Хелмана. Для теста шифрую пароль для архива. Это фбвгдеёж

Рис. 8. Взлом протокола методом «грубой силы» при различных ключах на шагах протокола

Проведя аналогичные математические вычисления, выяснится, что примерно за 20 минут можно пересмотреть все возможные варианты и при любом из вариантов найти параметры для шифрования и дешифровать исходное сообщение.

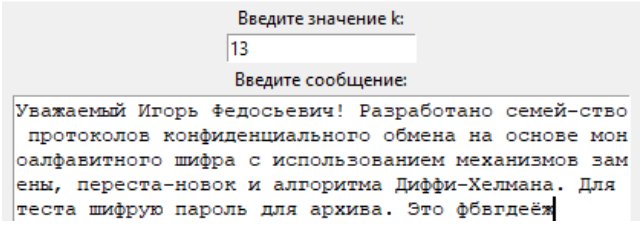
Таким образом, можно сделать вывод о том, что метод прямого подбора ключа по мощности алфавита, несмотря на свою простоту, остается ценным инструментом в арсенале криптоаналитиков. Он демонстрирует, как можно использовать математические свойства и характеристики текстов для восстановления зашифрованной информации. Этот подход подчеркивает важность глубокого понимания, что шифрование часто зависит не только от вычислительных мощностей, но и от интуиции и знаний о структуре языка.

Как было приведено выше, с помощью простого перебора можно найти нужное сообщение. На следующем этапе необходимо найти определённую последовательность символов, которую определим, заранее учитывая нормы русского языка. К примеру, последовательность из точки, пробела и заглавной буквы.

Наличие точки в конце предложения, за которой следует пробел и заглавная буква, служит чётким индикатором (признаком) границ предложений, что помогает создавать логическую структуру текста, позволяя анализировать его на уровне предложений. В языках с ясной пунктуацией, таких как русский, это особенно полезно, так как позволяет выделить смысловые единицы и облегчить понимание контекста. Использование данной последовательности помогает уменьшить количество возможных вариантов для дальнейшего анализа. После нахождения последовательности мы можем выделить предложения и анализировать их отдельно. Это позволяет сосредоточиться на меньших фрагментах текста, делая анализ более управляемым и менее подверженным ошибкам.

После того как допускаем предположения о некоторых буквах, мы можем проверить их на наличие в других предложениях, используя ту же последовательность как индикатор. Если текст начинает выглядеть осмысленно, получаем подтверждение правильности наших предположений. В противном случае мы можем корректировать наши гипотезы и повторять процесс. Данный метод является итеративным: после каждой проверки и корректировки мы можем возвращаться к анализу новых последовательностей, постепенно улучшая точность дешифрования. Это также открывает возможность для использования различных подходов, таких как анализ контекста и семантики, что может привести к более глубокому пониманию зашифрованного текста.

С этой целью рассмотрим вариант с учетом грамматических особенностей языков на примере шифрования, в котором используется один и тот же ключ для обоих шифров. Зашифрованный текст приведён на рисунке 9.



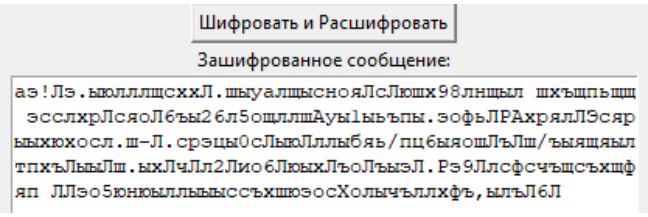


Рис. 9. Результаты шифрования и расшифрования

Написав небольшую программу для дешифровки с заданными условиями, в итоге получаем всего несколько вариантов, удовлетворяющих нашим требованиям (рис. 10).

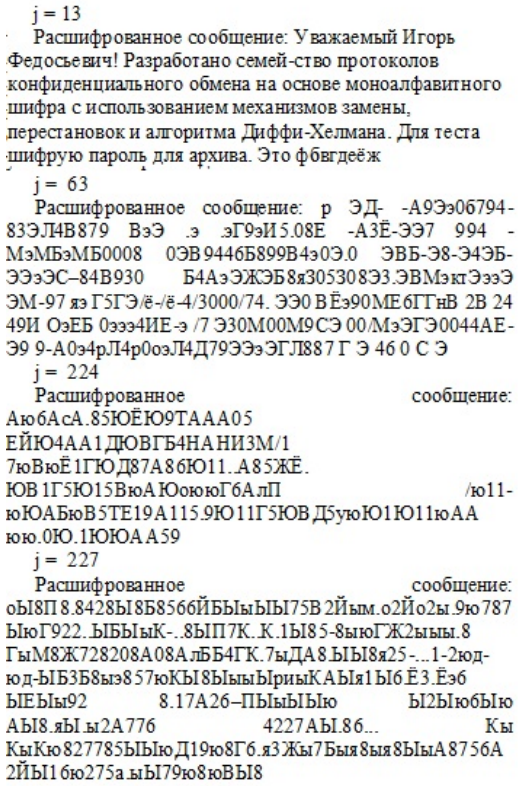


Рис. 10. Взлом протокола перебором общего ключа на шагах

Приведенный на рисунке 10 пример взлома протокола наглядно показывает, что всего для выбора было представлено четыре различных варианта ключа, в том числе один приемлемый.

Если же мы рассмотрим вариант, когда у нас два различных ключа шифрования, то в наихудшей ситуации будет увеличено количество вариантов в 5-6 раз соответственно.

Проанализировав всё вышеперечисленное, можно сделать вывод, что для взлома протокола, который использует моноалфавитный шифр замены и шифр перестановки, можно использовать не самые эффективные методы, такие как полный перебор без контекста. Это показывает слабость протокола, основанного на простых методах шифрования. Естественно, что при использовании контекстного поиска сложность нахождения правильных параметров для дешифровки сообщения многократно уменьшается.

Заключение

Эксперименты, выполненные на представленной в статье ИМ, приводят к следующим важным выводам.

Комбинированное использование моноалфавитного шифра замены и шифра перестановки может повысить уровень безопасности по сравнению с использованием каждого из этих методов по отдельности. Однако это не делает их безопасными от проанализированных в статье, видов атак. На примерах доказана уязвимость обоих методов к частотному анализу, что делает их ненадежными для защиты конфиденциальной информации. Злоумышленник, использующий методы криптоанализа, может восстановить оригинальное сообщение, если получит достаточно информации.

Следует отметить, что моноалфавитный шифр замены и шифр перестановки могут быть полезны для образовательных целей и понимания основ криптографии, однако они не подходят для защиты конфиденциальной информации в современных условиях, где угрозы становятся все более сложными и разнообразными.

Следовательно, для повышения безопасности рекомендуется использовать более сложные методы шифрования, которые обеспечивают «лавинный эффект». Поэтому в современных криптографических протоколах предпочтение отдается таким алгоритмам, как Магма, Кузнечик [8], AES [9]. Основанные на них криптографические протоколы обеспечивают высокий уровень безопасности и устойчивость к различным видам атак.

Также не стоит забывать полезность использования имитационного моделирования, которое позволило наглядно оценить эффективность примененных методов шифрования, проанализировать основанный на них протокол на устойчивость к различным видам атак. Данная ИМ позволила контролируемо определить параметры протокола и протестировать новые идеи и решения.

Литература

1. *Михалевич И.Ф., Абызов А.А., Архипов А.М., Басюк С.А.* и др. Имитационная модель SP-сети // REDS: Телекоммуникационные устройства и системы. 2023. № 2. С. 4-12.

2. *Ожиганов А.А*. Основы криптоанализа симметричных шифров: учебное пособие. СПб: СПбГУ ИТМО, 2008. 44 с. https://books.ifmo.ru/file/pdf/274.pdf.

3. *Григорьева Д.Р., Гареева Г.А., Басыров Р.Р*. Симметричные криптографичекие системы: учебно-методическое пособие по дисциплине «Информационная безопасность». Набережные Челны: Изд-во НЧИ КФУ, 2018. 30 с. https://kpfu.ru/staff_files/F105753572/EOR_KriptogrSistemy.pdf.

4. *Riham Altawy, Arm M. Yissef.* A Meet in the Middle Attack on Reduced Round Kuznyechik. October 2015 // IEICE Transactions on Fundamentals of Electronics Communications and Computer Sciences E98.A(10), pp. 2194-2198. http://dx.doi.org/10.1587/transfun.E98.A.2194.

5. *Manoj Ranjan Mishra, Jayaprakash Kar* A study on Diffie-Hellman key exchange protocols // International Journal of Pure and Applied Mathematics 114(2), (IJPAM). Vol. 114, No. 2 (2017). DOI:10.12732/ijpam.v114i2.2.

6. *Knuth D.E*. The Art of Computer Programming, Vol. 3, Sorting and Searching. Reading, MA.: Addison-Wesley, 1973. https://djvu.online/file/9d3Krt2uvWehr.

7. *Кнут Дональд Э*. Искусство программирования, том 3. Сортировка и поиск, 2-е изд.: Пер. с англ. М.: И.Д. Вильямс, 2007. 832 с.

8. ГОСТ Р 34.12-2015. Информационная технология. Криптографическая защита информации. Блочные шифры.

9. Advanced Encryption Standart (AES), NIST FIPS 197-upd1. https://nvlpubs.nist.gov/nistpubs/fips/nist.fips.197.pdf.

ОСОБЕННОСТИ ПРИМЕНЕНИЯ АЛГОРИТМА ГИБРИДНОЙ СОРТИРОВКИ

Герцев Константин Николаевич
МТУСИ, доцент кафедры СИТиС, к.т.н., Москва, Россия
kengaemu@yandex.ru

Литвинова Анастасия Михайловна
МТУСИ, студент группы М092401(75), Москва, Россия
litvinorable@mail.ru

Тремасова Лилия Андреевна
МТУСИ, ассистент кафедры СИТиС, к.т.н., Москва, Россия
l.a.tremasova@mtuci.ru

Гадасин Денис Вадимович
МТУСИ, доцент кафедры СИТиС, к.т.н., Москва, Россия
dengadiplom@mail.ru

Аннотация
Данная работа посвящена анализу алгоритмов сортировки данных в условиях интенсивного роста их объемов, сосредотачивая внимание на новом гибридном алгоритме Wolfsort, который сочетает преимущества рассмотренных в работе методов, являющихся его основой, а именно: быстрая сортировка, сортировка слиянием и поразрядная сортировка. Описаны ключевые этапы работы Wolfsort, включая анализ структуры данных, выбор оптимального метода для обработки отдельных сегментов массива и процесс объединения данных с использованием кэш-оптимизированных и многопоточных технологий. Основное внимание уделено эффективности алгоритма при обработке больших массивов и частично упорядоченных данных, а также его требованиям к памяти. Проведен сравнительный анализ алгоритмических сложностей различных методов, что подчеркивает преимущества Wolfsort в контексте современных вычислительных задач.

Ключевые слова
Обработка данных, большие данные, алгоритмы сортировки, WolfSort, оптимизация

Введение

В период с 2018 по 2024 наблюдается существенный рост мирового объема данных – с 33 зеттабайт до 150 зеттабайт, что подчеркивает необходимость в более продвинутых алгоритмах для их обработки и анализа. Согласно прогнозам, к 2025 году глобальный объём данных достигнет 175 зеттабайт, увеличившись более чем в пять раз по сравнению с показателями 2018 года (рис. 1).

Данный рост объясняется увеличением числа пользователей интернета, развитием технологий и повсеместным использованием цифровых устройств. Прогнозируется, что к 2025 году около 75% населения планеты будут ежедневно взаимодействовать с данными в интернете [1].

Исходя из роста объема информации, традиционные методы обработки данных становятся менее эффективными, что является большой проблемой, так как анализ такого большого количества информации требует современные подходы и алгоритмы, способные справляться с масштабируемостью и сложностью данных [2-5]. Стремительный рост объема данных подчеркивает необходимость в разработке и внедрении более продвинутых алгоритмов и технологий для их эффективной кластеризации, обработки и анализа, более совершенных систем обработки данных [6-7].

В основе обработки данных лежат алгоритмы сортировки, используемые в базах данных, машинном обучении, сетях и системах реального времени [8-10]. Сортировка позволяет экономить вычислительные ресурсы, снижать затраты времени и повышать производительность приложений, в том числе, за счёт балансировки нагрузки [11-13].

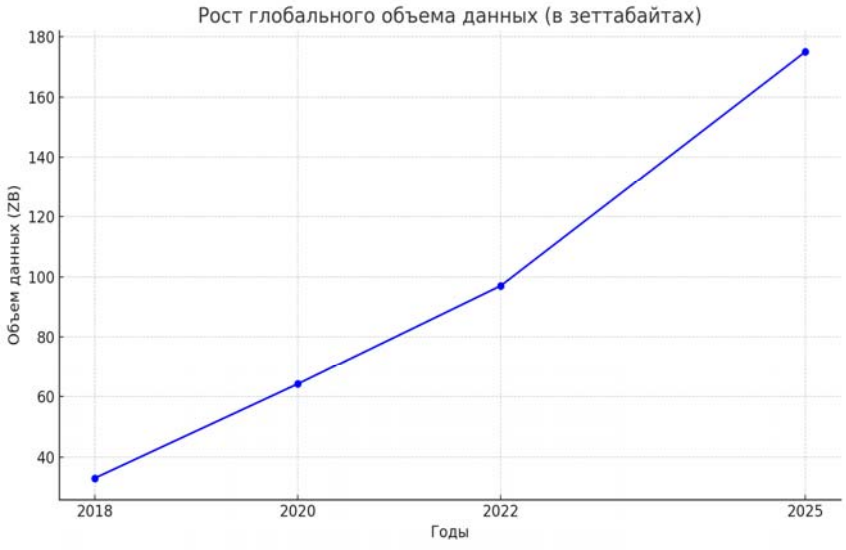


Рис. 1. Рост глобального объёма данных

Сортировка представляет собой переход данных от начального состояния к упорядоченному, что является одним из возможных их состояний [14-15]. В контексте работы с данными упорядоченность – это не только способ структурировать данные, но и этап, подготавливающий их для дальнейших операций, таких как поиск, агрегация и фильтрация. Применяя сортировку, мы приводим данные в последовательность, которая значительно упрощает и ускоряет последующую обработку. За счёт такого упрощения получается увеличить производительность системы с учётом растущих объёмов данных.

Каждый алгоритм имеет свою вычислительную сложность, которая влияет на его производительность в разного рода задачах. Для небольших объёмов данных достаточно эффективно можно использовать традиционные алгоритмы, которые обычно изучаются при освоении основ алгоритмов и структур данных, например пузырьковая сортировка [16]. Как бы то ни было, с ростом объема данных сложность задачи возрастает, что требует применения более продвинутых методов, например, сортировки слиянием или поразрядной сортировки. Всё более широкое применение находят гибридные методы сортировки, сочетающие преимущества нескольких подходов. Одним из таких алгоритмов является Wolfsort, который представляет собой адаптивный гибридный метод сортировки. В данной работе будет исследована применимость алгоритма Wolfsort к сортировке больших наборов чисел.

Базовые алгоритмы

Классические подходы, такие как сортировка слиянием, быстрая сортировка и поразрядная сортировка, решают одну и ту же задачу разными способами, используя при этом уникальные стратегии обработки данных, которые имеют свои сильные и слабые стороны, а их выбор зависит от структуры данных и требований к производительности, а также сохранности данных.

Сортировка слиянием (Merge Sort) – это алгоритм, основанный на принципе, который рекурсивно делит массив на две части, сортирует каждую из них, и, после полного разбиения, объединяет обратно в один массив [17]. Основная идея заключается в том, что объединить два отсортированных массива проще, чем отсортировать их с нуля. Алгоритм состоит из двух этапов: разбиение массива на подмассивы до тех пор, пока каждый из них не станет размером 1, и объединение подмассивов в один отсортированный массив. Рисунок 2 отображает алгоритм обработки входных данных, когда сам массив уже разбит на массивы из одного элемента.

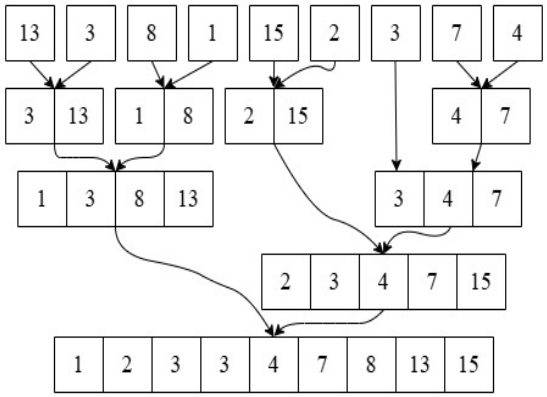


Рис. 2. Пример обработки входных данных алгоритмом Merge Sort

Массив [13, 3, 8, 1, 15, 2, 3, 7, 4] делится на две части. Затем каждая часть снова делится, и так далее, пока не останутся массивы из одного элемента: [13], [3], [8], [1], [15], [2], [3], [7], [4]. После разделения массивов начинается этап объединения, на котором происходит сравнение элементов и формируется отсортированный массив. Сравниваются элементы 13 и 3. В результирующий массив вперёд добавляется меньший элемент и получается массив [3, 13]. То же самое с элементами 8 и 1: [1, 8]. Затем сравниваются первые элементы обеих частей (3 и 1). В результирующий массив добавляется меньший элемент (1), затем сравниваются следующие элементы: [1, 3, 8, 13].

Тот же процесс повторяется для правой части массива: [15] и [2] → [2, 15], [3] и [7, 4] → [3, 4, 7], [2, 15] и [3, 4, 7] → [2, 3, 4, 7, 15]. Объединяются два отсортированных подмассива: [1, 3, 8, 13] и [2, 3, 4, 7, 15]. Результат: [1, 2, 3, 3, 4, 7, 8, 13, 15]

На основе вышеизложенной информации посчитана алгоритмическая сложность алгоритма слияния с использованием принципов большой нотации (Big-O notation).

Количество сравнений при объединении двух отсортированных массивов можно выразить как в формуле 1:

$$C(n) = n - 1 \tag{1}$$

где n – количество элементов в объединяемом массиве.

Сложность алгоритма сортировки слиянием можно выразить с помощью рекуррентного уравнения, показанного в формуле 2:

$$T(n) = 2T \cdot (\frac{n}{2}) + O(n \cdot \log(n)) \tag{2}$$

где $T(n)$ – время выполнения для массива длины n; $2T(\frac{n}{2})$ – время, затраченное на сортировку двух половин массива; $O(n)$ – время, затраченное на объединение двух половин.

Таким образом, сортировка слиянием имеет гарантированную логарифмическую сложность и работает результативно даже на больших массивах. Плюс к этому, она сохраняет порядок одинаковых элементов, но при этом требует дополнительной памяти для временных массивов.

Алгоритм быстрой сортировки (Quick Sort) считается одним из самых популярных и наиболее эффективных алгоритмов. О работает на основе стратегии «разделяй и властвуй», где массив разделяется относительно опорного элемента на две группы: элементы меньше опорного и элементы больше опорного. Эти части затем рекурсивно сортируются тем же методом, что демонстрируется на рисунке 3 [14].

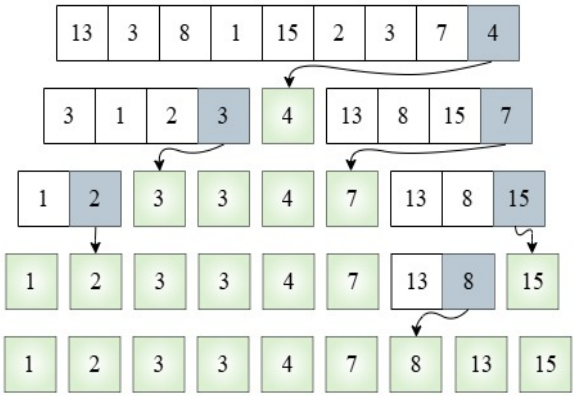


Рис. 3. Пример обработки входных данных алгоритмом Quick Sort

Изначально имеется массив чисел: [13, 3, 8, 1, 15, 2, 3, 7, 4]. Алгоритм выбирает опорный элемент, в данном примере опорным элементом всегда выбирается последний элемент подмассива, но это не обязательное условие, опорным элементом можно выбирать любое число массива. На первом шаге опорным элементом является 4. Массив делится на две части: слева располагаются элементы меньше опорного – 3, 1, 2, 3, а справа больше – 13, 8, 15, 6.

Затем алгоритм рекурсивно применяется к левой части. Опорным элементом для этого подмассива становится последний элемент – 3. Массив снова делится на две части по тому же принципу: слева остаются элементы, меньшие или равные 3 – [1, 2], а справа элементы, большие 3 – [3]. Далее алгоритм применяется к подмассиву [1, 2]. Опорным элементом выбирается 2. Массив делится на две части: слева располагаются элемент [1], а справа – пустой подмассив (все элементы больше 2 отсутствуют). Так как подмассивы [1] и [2] содержат только один элемент, они считаются отсортированными. Таким образом, левая часть полностью отсортирована.

Теперь алгоритм переходит к правой части исходного массива – 13, 8, 15, 7. Опорным элементом выбрана 7. По аналогии с левой половиной, происходит сортировка в правой. Массив делится на две части: слева остаются элементы, меньшие или равные 7 (пустой подмассив), а справа элементы, большие 7, то есть [13, 8, 15]. Это демонстрирует неудачный выбор опорного элемента, что не влияет на корректность алгоритма, но сильно снижает его эффективность. Далее вновь происходит неудачный выбор опорного элемента, так как по алгоритму выпало самое большое число массива. В контексте данного примера, это не сильно замедлит работу сортировки, но при работе с большими данными это отняло бы много времени. В конечном счёте рекурсивно правая часть отсортирована так же, как и левая. На последнем шаге объединяются отсортированные подмассивы. Левый подмассив [1, 2, 3, 3], опорный элемент 4 и правый подмассив [7, 8, 13, 15] объединяются в итоговый отсортированный массив: [1, 2, 3, 3, 4, 7, 8, 13, 15].

В соответствии с этапами обработки входного массива, рассчитаем алгоритмическую сложность быстрой сортировки.

Быстрая сортировка имеет разную сложность в зависимости от структуры данных. Формула 3 демонстрирует среднюю алгоритмическую сложность быстрой сортировки.

$$T(n) = T(k) + T(n-k-1) + O(n) \tag{3}$$

где $T(n)$ – время выполнения для массива длины; k – количество элементов в одной из частей после разделения; $O(n)$ – время на разделение массива.

В ситуации наихудшего сценария, когда сложность достигает $O(n^2)$, возникает, когда массив всякий раз разделяется на крайне неравномерные части (например, как вышло в разобранном примере в правой части, если в качестве опорного элемента каждый раз выбирается минимальный или максимальный элемент из возможных).

В противоположность этому, наилучший случай достигается, когда массив неизменно делится на две равные части, в таком случае сложность сокращается до $O(n \cdot \log(n))$.

Среднее количество сравнений в быстрой сортировке можно оценить как показано в формуле (4):

$$C(\mathrm{n}) = 2 \cdot \mathrm{n} \cdot \log(\mathrm{n}) \tag{4}$$

где n – количество элементов в массиве.

Таким образом, этот алгоритм действительно быстрый на практике, особенно для случайных массивов, требует мало дополнительной памяти, но при этом нестабильный, так как не сохраняет порядок одинаковых элементов и очень сильно зависит от выбора опорного элемента.

Поразрядная сортировка (Radix Sort) – это алгоритм, который вместо того, чтобы сравнивать элементы, как это делают классические алгоритмы сортировки, группирует числа по значениям их разрядов и обрабатывает их последовательно. Radix Sort работает следующим образом: создается временный массив для хранения количества вхождений каждой цифры, по этому массиву рассчитываются позиции элементов в отсортированном массиве, после чего элементы размещаются в правильном порядке [18-20]. Рисунок 4 демонстрирует этапы обработки входных данных алгоритмом Radix Sort.

На первом этапе происходит сортировка чисел по младшему разряду (единицам). Числа группируются в зависимости от значения этой цифры, например, числа с тройкой в младшем разряде (3, 13) оказываются в одной группе. После выполнения этого шага получается промежуточный порядок, где числа частично отсортированы по единичному разряду.

На втором этапе алгоритм переходит к следующему разряду – десяткам. Сортировка теперь производится по второй цифре (десяткам) каждого числа. Те числа, у которых десятки равны нулю, попадают в начало списка, а затем идут числа с десятками 1, 2 и так далее. После выполнения этого шага числа становятся упорядоченными уже по первым двум разрядам.

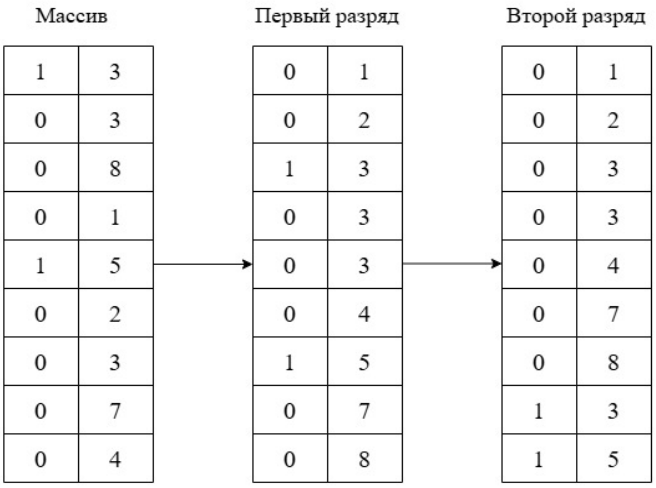


Рис. 4. Этапы обработки входных данных алгоритмом Radix Sort

На третьем этапе производится сортировка по следующему разряду. Поскольку в данном примере максимальное число – 15, третий разряд не участвует, и итоговая последовательность остаётся неизменной.

В результате все числа оказываются отсортированными в возрастающем порядке. Алгоритм Radix Sort особенно эффективен для чисел с небольшим количеством разрядов, так как каждый проход выполняет линейную сортировку.

На основе вышесказанной информации, произведём расчет алгоритмической сложности алгоритма Radix Sort. Поразрядная сортировка выполняет d проходов по массиву, где d – это количество разрядов в наибольшем числе. На каждом проходе выполняется сортировка методом подсчета, которая имеет сложность $O(n+k)$. Таким образом, общее время выполнения алгоритма рассчитывается как в формуле (5):

$$T(n) = O(d \cdot (n + k)) \tag{5}$$

где n – количество элементов в массиве; k – диапазон возможных значений для каждого разряда.

Таким образом, сортировка слиянием характеризуется линейной сложностью, что делает её более продуктивной при работе с большим набором данных с ограниченным диапазоном значений. Но алгоритм не универсален, он применим только для числовых данных фиксированной длины, а также требует дополнительной памяти для временных массивов и становится менее эффективным при большом количестве разрядов.

Рассмотренные алгоритмы демонстрируют достойные результаты при работе с небольшими наборами данных. Обращая внимание на алгоритмическую сложность каждого из вышеописанных алгоритмов, важно отметить, что по мере увеличения размеров этих данных, производительность системы будет заметно снижаться. Поэтому существует необходимость реализации новых алгоритмов с упором на обработку огромных наборов данных, при котором скорость их обработки в системе будет иметь минимальные потери в производительности.

Модель алгоритма Wolfsort

Современным решением проблемы подготовки больших наборов данных является алгоритм сортировки Wolfsort. Как говорилось ранее, Wolfsort является гибридным алгоритмом, объединяющий в себе лучшие практики классических алгоритмов, компенсируя при этом проблемы с производительностью. Алгоритм был разработан для оптимальной обработки массивов среднего и большого размера, особенно тех, которые содержат как случайные, так и частично упорядоченные данные.

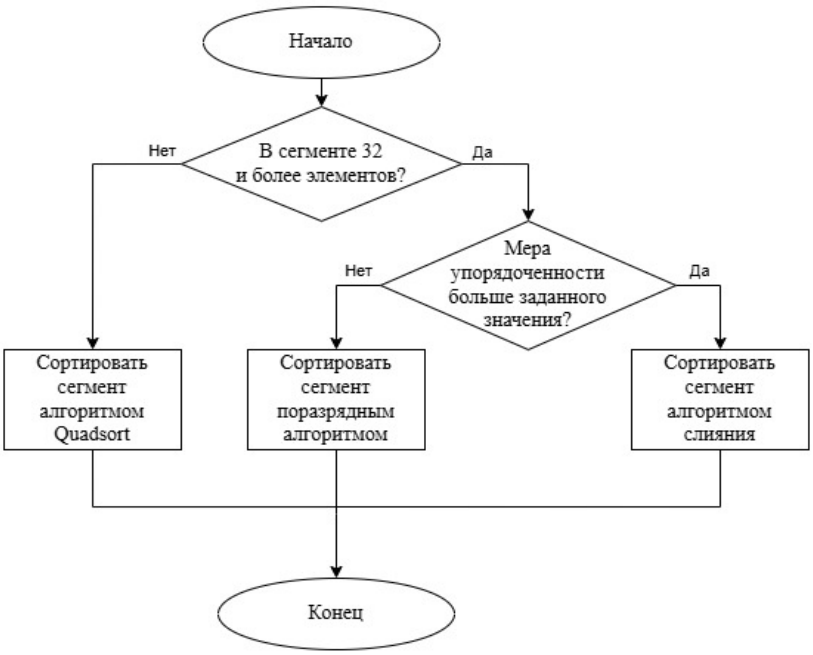


Рис. 5. Алгоритм работы WolfSort

Он сочетает адаптивное разбиение массива и быструю обработку подмассивов, что позволяет ему минимизировать количество операций и эффективно использовать ресурсы процессора и кэш-памяти. Рассмотрим основные этапы работы этого алгоритма с сегментами данных, схема показана на рисунке 5.

1. На первом этапе WolfSort анализирует структуру входного массива, чтобы определить, насколько он упорядочен. Алгоритм делит массив на несколько сегментов и вычисляет меру упорядоченности каждого сегмента. По умолчанию WolfSort делит массив на четыре сегмента. Это деление может быть изменено, но стандартный подход – использовать именно четыре сегмента для оптимального баланса между производительностью и анализом структуры данных. Обозначим массив как A длины n. Формула 6 описывает начальное состояние массива A:

$$A = [a_1, a_2, ..., a_n] \tag{6}$$

Разделим массив на четыре сегмента. Формула (7) описывает результат деления массива на сегменты:

$$A = [S_1, S_2, S_3, S_4] \tag{7}$$

где каждый сегмент имеет длину примерно, как показано в формуле (8):

$$\left| S_i \right| = \frac{n}{4} \tag{8}$$

Для каждого сегмента WolfSort вычисляет меру упорядоченности. Пусть $O(S_i)$ – функция, которая оценивает степень упорядоченности сегмента S_i:

- если сегмент почти отсортирован, $O(S_i)$ будет близко к 1.
- если сегмент хаотичный, $O(S_i)$ будет близко к 0.

Пример формулы оценки упорядоченности:
$O(S_i)$ = количество упорядоченных пар/общее количество пар.

2. На втором этапе происходит выбор алгоритма сортировки для каждого сегмента. На основе меры упорядоченности $O(S_i)$ WolfSort принимает решение, какой алгоритм использовать для сортировки сегмента. Для частично отсортированных данных ($O(S_i) \geq 0.75$) выбор упадёт на алгоритм слияния, а для случайных - поразрядная.

Для небольших сегментов (менее 32 элементов) алгоритм использует Quadsort по умолчанию, так как он наиболее эффективен для сортировки небольших массивов.

3. После выбора подходящего алгоритма каждый сегмент сортируется независимо. Сегменты могут сортироваться параллельно, что ускоряет процесс сортировки на многопроцессорных системах. Каждый сегмент обрабатывается локально, что снижает затраты на операции копирования и уменьшает задержки при доступе к памяти.

4. На последнем этапе WolfSort объединяет ранее отсортированные сегменты в один массив. Этот процесс объединения играет ключевую роль в эффективности алгоритма. В отличие от стандартного подхода в сортировке слиянием, WolfSort оптимизирует процесс объединения таким образом, чтобы минимизировать накладные расходы на копирование данных и оптимизированно использовать кэш-память процессора.

WolfSort использует многопоточное объединение, что позволяет одновременно объединять несколько сегментов. Алгоритм инициализирует временный массив для хранения объединенных данных, каждый сегмент получает указатель, который отслеживает текущую позицию в сегменте. На каждом шаге сравниваются текущие элементы из каждого сегмента, и меньший элемент добавляется в временный массив. Этот процесс повторяется, пока все сегменты не будут объединены в один массив.

WolfSort делает акцент на оптимизации использования кэш-памяти процессора во время объединения сегментов. Это достигается за счет:

- Блочного копирования данных, которые хорошо вписываются в кэш-память;
- Сокращения операций копирования, так как вместо копирования каждого элемента по одному, WolfSort старается копировать последовательные блоки данных за один раз;
- Параллельной обработки, если аппаратное обеспечение поддерживает многопоточность.

Математическое описание расчёта потребления памяти алгоритмом продемонстрировано в формуле 9:

$$Memory = 4 \cdot n + \frac{n}{4} \tag{9}$$

где n – количество элементов в массиве.

За счёт этого, временная сложность алгоритма в лучшем случае, когда входные данные уже почти отсортированы, достигает $O(n)$, за счёт способности эффективно распознавать упорядоченность данных и минимизировать количество операций. Алгоритм демонстрирует временную сложность $O(n \cdot \log(n))$ в среднем и худшем случаях.

Заключение

Исходя из вышеперечисленного, WolfSort – это мощный и адаптивный алгоритм сортировки, который сочетает лучшие черты его составных алгоритмов. Его способность эффективно обрабатывать как случайные, так и частично упорядоченные массивы делает его оптимальным выбором для множества практических приложений. Благодаря гибридной природе, адаптивному разбиению и оптимизации использования памяти WolfSort обеспечивает баланс между стабильностью, скоростью и эффективностью.

Но при этом, WolfSort – это не "in-place" алгоритм. Для работы ему необходимо выделить больше памяти в сравнении с классическими алгоритмами. Данная память ответственна за хранение отсортированных сегментов и временных массивов. Таким образом, работа алгоритма является менее эффективной в рамках ограниченной доступной памяти (например, встраиваемые системы или системы с жесткими ограничениями по оперативной памяти), и WolfSort будет неподходящим выбором.

Плюс ко всему, WolfSort зависит от структуры входных данных. Он наиболее эффективен на частично упорядоченных массивах, но на полностью случайных массивах его производительность может быть сравнима с традиционными алгоритмами, на которых он основан.

WolfSort стоит использовать в случаях, когда размер массива средний или большой (1000+ элементов), сам массив частично упорядочен, при этом требуется стабильная сортировка и доступно достаточное количество оперативной памяти.

Литература

1. *David Reinsel, John Gantz, John Rynding*. The Digitization of the World from Edge to Core. Data Age 2025. 2020.

2. *Гадасин Д.В., Бессолицын А.Д., Гадасин Д.Д.* Оценка качества данных информационных систем // DSPA: Вопросы применения цифровой обработки сигналов. 2024. Т. 14, № 2. С. 4-12. EDN GYIWJU

3. *Гадасин Д.В., Лисиненко Е.К., Юсифов Э.С., Савин В.А*. Оценка регрессионных моделей исходя из показателей качества / // Системы синхронизации, формирования и обработки сигналов. 2024. Т. 15, № 1. С. 4-16. EDN CSWKOE

4. *Шведов А.В., Гадасин Д.В., Коровушкина В.М., Мелькова Е.К*. Интеллектуальное тестирование как способ повышения качества информационной системы // REDS: Телекоммуникационные устройства и системы. 2022. Т. 12, № 2. С. 43-52. EDN GOLZGE

5. *Гадасин Д.В., Михайлов М.Р., Чернышов Д.В.* Определение алгоритма структурирования текстовых данных // REDS: Телекоммуникационные устройства и системы. 2024. Т. 14, № 1. С. 4-11. EDN GLAEQF

6. *Гадасин Д.В., Золотарева П.Ю., Тремасова Л.А.* Влияние кластеризации при обработке сырых данных // Системы синхронизации, формирования и обработки сигналов. 2024. Т. 15, № 3. С. 10-19. EDN JQIPHX

7. *Тремасова Л.А., Первухина А.А., Гадасин Д.В*. Использование методов Косарайю и k-средних для формирования кластеров // Электросвязь. 2024. № 9. С. 47-55. DOI 10.34832/ELSV.2024.58.9.007. EDN DOZTZK

8. *Gadasin D.V., Panteleeva K.A., Shvedov A.V., Maklachkova V.V*. Using formants for human speech recognition by artificial intelligence // Systems of Signal Synchronization, Generating and Processing in Telecommunications. 2023. Vol. 6, No. 1, pp. 100-106. DOI 10.1109/SYNCHROINFO57872.2023.10178431. EDN XTXWOI

9. *Гадасин Д.В., Шведов А.В., Кузин И.А.* Трехмерная реконструкции объекта по одному изображению с использованием глубоких свёрточных нейронных сетей // T-Comm: Телекоммуникации и транспорт. 2022. Т. 16, № 7. С. 29-35. DOI 10.36724/2072-8735-2022-16-7-29-35. EDN YTLCNW

10. *Маклачкова В.В., Гадасин Д.В., Литвинов Д.С., Кувшинов М.И*. Проектирование системы поддержки принятия решений по подбору подарка // DSPA: Вопросы применения цифровой обработки сигналов. 2024. Т. 14, № 1. С. 51-61. EDN CVTSLG

11. *Гадасин Д.В., Смальков Н.А., Кузин И.А.* Использование метода роя частиц для балансировки нагрузки в сетях Интернета вещей // Системы синхронизации, формирования и обработки сигналов. 2022. Т. 13, № 2. С. 17-23. EDN LIUWNT

12. *Гадасин Д.В., Шведов А.В*. Применение транспортной задачи для балансировки нагрузки в условиях нечеткости исходных данных // T-Comm: Телекоммуникации и транспорт. 2024. Т. 18, № 1. С. 13-20. DOI 10.36724/2072-8735-2024-18-1-13-20. EDN WKNPIX

13. *Гадасин Д.В., Андриянова А.К., Тремасов Л.А*. Определение нечеткости поискового запроса через множество аксиом объекта а // Технологии информационного общества: Сборник трудов XVII Международной отраслевой научно-технической конференции, Москва, 02-03 марта 2023 г. М.: Издательский дом Медиа Паблишер, 2023. С. 132-134. EDN DWJUTE

14. *Гадасин Д.В.* Построение бинарного дерева минимальной цены // T-Comm: Телекоммуникации и транспорт. 2024. Т. 18, № 11. С. 38-44. DOI 10.36724/2072-8735-2024-18-11-38-44. EDN GMCEWG

15. *Gadasin D.V., Shvedov A.V., Kuzin I.A.* A model for representing the color and depth metric characteristics of objects in an image // 2021 Systems of Signal Synchronization, Generating and Processing in Telecommunications, SYNCHROINFO 2021 – Conference Proceedings, Svetlogorsk, Kaliningrad Region, 30 июня – 02 2021 г. Svetlogorsk, Kaliningrad Region, 2021. P. 9488349. DOI 10.1109/SYNCHROINFO51390.2021.9488349. EDN YAYZVP

16. *Edwards J., McNamee J.* A Comparison of sorting algorithms. ACM Computing Surveys. 1970. Vol. 2, No. 3, pp. 195-213.

17. *Petersen P., Thorup M*. Quicksort is optimal for many equal keys. SIAM Journal on Computing. 2007. Vol. 40, No. 1, pp. 1-17.

18. *Gill S.K., Singh V.P., Sharma P., Kumar D*. Comparative study of various sorting algorithms // International Journal of Advanced Studies of Scientific Research. 2019. Т. 4, № 1. С. 1-7.

19. *Гадасин Д.В., Вакурин И.С., Тремасова Л.А*. Алгоритм распределения данных между системами хранения на основе свойства самоподобия // Электросвязь. 2024. № 4. С. 44-50. DOI 10.34832/ELSV.2024.53.4.015. EDN BRSLCL

20. *Гадасин Д.В., Шведов А.В., Вакурин И.С*. Определение семантической близости текстов с использованием алгоритма сравнения сущности графов // REDS: Телекоммуникационные устройства и системы. 2022. Т. 12, № 4. С. 11-19. EDN PVJKQJ