ВЕСТНИК
НОВОСИБИРСКОГО ГОСУДАРСТВЕННОГО УНИВЕРСИТЕТА

Научный журнал
Основан в ноябре 1999 года

Серия: Информационные технологии 2023. Том 21, № 2

СОДЕРЖАНИЕ

ISSN 1818-7900 (Print). ISSN 2410-0420 (Online)
Вестник НГУ. Серия: Информационные технологии. 2023. Том 21, № 2
Vestnik NSU. Series: Information Technologies, 2023, vol. 21, no. 2

V E S T N I K
NOVOSIBIRSK STATE UNIVERSITY

Scientific Journal
Since 1999, November
In Russan

Series: Information Technologies 2023. Volume 21, № 2

CONTENTS

ISSN 1818-7900 (Print). ISSN 2410-0420 (Online)
Вестник НГУ. Серия: Информационные технологии. 2023. Том 21, № 2
Vestnik NSU. Series: Information Technologies, 2023, vol. 21, no. 2

The journal is published quarterly in Russian since 1999
by Novosibirsk State University Press

The address for correspondence
Faculty of Information Technologies, Novosibirsk State University
1 Pirogov Street, Novosibirsk, 630090, Russia
Tel. +7 (383) 363 42 46

E-mail address: inftech@vestnik.nsu.ru
On-line version: http://elibrary.ru

ISSN 1818-7900 (Print). ISSN 2410-0420 (Online)
Вестник НГУ. Серия: Информационные технологии. 2023. Том 21, № 2
Vestnik NSU. Series: Information Technologies, 2023, vol. 21, no. 2

Научная статья

УДК 004.434
DOI 10.25205/1818-7900-2023-21-2-5-17

К вопросу о разрешении проблемы конкурирующих процессов в процесс-ориентированном программировании

Дмитрий Андреевич Пермяшкин

Институт автоматики и электрометрии СО РАН
Новосибирск, Россия

d.permiashkin@g.nsu.ru, https://orcid.org/0009-0004-6045-3228

Аннотация

В современном мире значительная часть фабрик и промышленных производств уже управляются программируемыми микроконтроллерами и число такого рода производств непрерывно растет. Данный процесс тесно взаимосвязан с идеями Индустрии 4.0, а если быть точнее – с идеей полной автоматизации производственных процессов для облегчения принятия решений человеку. И одновременно уменьшения числа принимаемых человеком решений вплоть до полного отсутствия человека в роли принимающего решения. Для этого требуются управляющие алгоритмы, которые должны работать по событиям, учитывать наличие и тесный контакт с внешней средой и иметь повышенные требования к устойчивости к внутренним отказам и отказам оборудования. Разработанная в Институте автоматики и электрометрии СО РАН процесс-ориентированная парадигма программирования, рассматриваемая в данной статье, прекрасно подходит для разработки управляющих алгоритмов такого рода. Данная парадигма учитывает тот факт, что производственный процесс представляет собой очень большое число одновременно работающих процессов, тесно связанных с элементами реального мира. Отсюда вытекает необходимость решения проблемы конкурирующих процессов для обеспечения требуемой степени устойчивости к отказам. В данной статье будет поставлена и проанализирована проблема конкурирующих процессов. Для этого в работе был произведен анализ существующих решений проблемы конфликтов при параллельной работе произвольного числа процессов с целью анализа применимости данных методов для процесс-ориентированных программ или же возможности адаптации некоторых методов. В результате будет получен ответ на вопрос, насколько эффективно процесс-ориентированная парадигма позволяет решать конфликты при параллельном исполнении программы и насколько сильно удовлетворяет требованию отказоустойчивости.

Ключевые слова

индустриальное программирование, параллельные программы, процесс-ориентированное программирование, процесс-ориентированные языки

Для цитирования

Пермяшкин Д. А. К вопросу о разрешении проблемы конкурирующих процессов в процесс-ориентированном программировании // Вестник НГУ. Серия: Информационные технологии. 2023. Т. 21, № 2. С. 5–17. DOI 10.25205/1818-7900-2023-21-2-5-17

On Solving Concurrent Process Problem in Process-Oriented Programs

Dmitry A. Permiashkin

Institute of Automation and Electrometry SB RAS
Novosibirsk, Russian Federation

d.permiashkin@g.nsu.ru, https://orcid.org/0009-0004-6045-3228

Abstract

Prevailing portion of the factories and manufacturers are controlled by programming microcontrollers in the modern world. And the portion keeps growing which is closely tied to processes of the Fourth Industrial Revolution. Precisely, with an idea of fully automated manufactures to help humans to make less decisions and make them faster. Or to exclude humans from the decision making process at all. Due to that, there is a need for the controlling algorithms which should react to the different events, be aware of the external world and be tolerant to both internal and hardware failures. There is a process-oriented paradigm which was developed in Institute of Automation and Electrometry SB RAS and suits perfectly for automatization of such algorithms. This is achieved by splitting the algorithm into huge amounts of the small parallel processes highly tied to the elements of the real world. Which is how real processes on real manufactures work. This is where the conflicts during concurrent programming appear. And because there is a fault tolerance requirement there is a need to solve those conflicts. This work presents the analysis of already existing solutions to the conflicts during concurrent programming with the goal of either reusing those solutions in process-oriented programming or adapting them to it. As a result, there is an answer on how effective the process-oriented paradigm is in solving those kinds of conflicts and how fault tolerant those programs are.

Keywords

industrial programming, concurrent programming, process-oriented programming, process-oriented language

For citation

Permiashkin D. A. On Solving Concurrent Process Problem in Process-Oriented Programs. *Vestnik NSU. Series: Information Technologies*, 2023, vol. 21, no. 2, pp. 5–17. DOI 10.25205/1818-7900-2023-21-2-5-17

Введение

В современном мире активно идет процесс автоматизации. Автоматизируется складской учет, производства и даже уже сам процесс автоматизации. Для автоматизации таких процессов очень важна возможность параллельного исполнения. В связи с этим в данной области популярны системы на основе программируемых логических контроллеров (ПЛК), которые не только позволяют поддерживать параллелизм на очень высоком уровне, но также требуют меньше электричества, имеют меньшее тепловыделение и более устойчивы к воздействию внешней среды (тепловому и электрическому). Для облегчения разработки самих контроллеров и программного обеспечения был разработан стандарт IEC 61131 в 10 частях, из которых самой важной частью является третья, описывающий стандарты языков для программирования ПЛК.

Третья и последняя версия стандарта IEC 61131-3 вышла в 2013 году [1], что не кажется недостатком на первый взгляд. Но тут встает вопрос, насколько успели повлиять на стандарт идеи четвертой индустриальной революции, или же Индустрии 4.0, учитывая вынужденную медлительность разработки любого стандарта. Данный вопрос очень важен, поскольку в ключевых темах находится сильная взаимосвязь всех элементов и активная помощь в принятии решений человеку с одновременным исключением необходимости принятия решения человеком в как можно большей части задач. В связи с этим уже ведется разработка новых парадигм программирования и языков на них с учетом Индустрии 4.0. Одним из основных языков, реализующих идеи из Индустрии 4.0, является рассматриваемая в данной статье процесс-ориентированная парадигма программирования, разработанная в Институте автоматики и электрометрии СО РАН и уже проверенная на практике при автоматизации производств [2, 3, 4].

Этот подход обеспечивает комфортное создание и сопровождение управляющих алгоритмов, отличие которых от вычислительных алгоритмов заключается в наличии внешней среды, необходимости работать по событиям, в частности, временным, параллелизму, и повышенным требованиям по надежности, в частности, требованиям устойчивости к внутренним отказам и отказам оборудования.

Для такого рода алгоритмов очень важна возможность параллельного исполнения. И не просто возможность выполнения нескольких никак не связанных процессов, но возможность выполнения большого числа процессов, связанных между собой разделяемыми ресурсами – элементами реального мира. Поэтому при автоматизации такого рода всегда встает вопрос о разрешении конфликтов, возникающих при параллельных вычислениях.

Данная сфера уже достаточно активно изучается, и первый алгоритм, решающий задачу о конфликтах при параллельных вычислениях при произвольном числе параллельных программ, был опубликован в 1965 году. Однако в силу специфики управляющих алгоритмов методы, разработанные в рамках теории параллельных вычислений, часто не могут быть использованы напрямую и нуждаются в адаптации. Основные причины – относительно низкая вычислительная мощность узлов и необходимость обеспечения устойчивости в случае отказов в системе. Таким образом, алгоритмы разрешения конфликтов должны иметь низкие требования к ресурсам и быть способны в случае отказа приводить оборудование в безопасное состояние. Например, при автоматизации процесса выращивания монокристаллического кремния разрушение тигля с расплавленным кремнием должно вызывать отключение нагревателя и перевод тигля в крайнее нижнее положение [5].

Таким образом, эффективная и надежная реализация межпроцессного взаимодействия в рамках процесс-ориентированного программирования является актуальной задачей.

В статье анализируются существующие подходы к разрешению конфликтов в параллельных программах и распределенных системах управления, рассматривается специфика межпроцессного взаимодействия в процесс-ориентированной модели программы, обсуждаются применимость существующих подходов к организации межпроцессного взаимодействия в рамках процесс-ориентированной парадигмы.

Обзор существующих подходов к разрешению конфликтов в параллельных программах

Для целей решения конфликтов параллельная программа состоит из двух частей – процессы и переменные. Процесс – это последовательный набор подписанных инструкций (или же операций). Переменные содержат какие-либо значения и могут быть доступны либо всем процессам (публичная), либо только внутри одного конкретного процесса (частная или же локальная). При этом у любого процесса есть счетчик команд – особая переменная, которая указывает на текущую выполняемую операцию в процессе. Процесс выполняет некоторый бесконечный цикл из двух секций:

- некритическая секция, где любая операция изменяет только частные переменные;
- критическая секция, где операции могут изменять публичные и частные переменные.

Введем понятие примитива – последовательного набора операций. Примитив будет атомарным, если гарантируется, что в момент исполнения примитива любая другая операция не будет разрешенной. Поэтому далее атомарный примитив будет рассматриваться как одна операция.

Состоянием назовем отображение некоторых значений во все переменные программы. Состояние может быть начальным – возможным при начале исполнения программы. Назовем операцию s разрешенной, если можно перейти из состояния t в состояние u после выполнения операции s. История – это последовательность из состояний и операций, где любая операция

является разрешенной. А конфликт – момент в истории, где изменение очередности выполнения разрешенных операций меняет итоговое состояние.

В основном стандартные подходы к решению можно разделить на два класса:

1. Использующие взаимное исключение:
– на основе общей памяти,
– при помощи сообщений;
2. Не использующие взаимное исключение.

Первый алгоритм для разрешения конфликтов на основе взаимного исключения был опубликован Дийкстрой в 1965 году в статье [6] и расширяет критическую секцию, разделяя ее на секцию входа, саму критическую секцию и секцию выхода. Далее к секциям предъявляются дополнительные требования:

- Процесс не может остановиться в критической секции.
- Любая инструкция либо в любом состоянии не является разрешенной, либо она исполнилась (требование честности для истории).
- Секции входа и выхода должны гарантировать, что критическая секция исполняется максимум одним процессом в любой момент времени (требование исключения).
- Секции входа и выхода должны гарантировать, что если какой-то процесс находится в критической секции, то какой-то другой процесс рано или поздно будет в критической секции (требование свободы от блокирования).

Статья [6] породила много обсуждений, из которых стоит упомянуть статью Кнута [7], где было введено более строгая версия требования свободы от блокирования:

- Секции входа и выхода должны гарантировать, что если какой-то процесс находится в секции входа в критическую секцию, то этот процесс рано или поздно будет в критической секции.

Дальнейшие работы в основном не вводили каких-то новых требований, а сфокусировались на решении самой проблемы в рамках этих требований.

Алгоритмы взаимного исключения на основе общей памяти. Задача исключения в данных алгоритмах решается хранением в общей памяти переменной или переменных, описывающих состояние критической секции. При этом любая переменная, кроме счетчика команд, может одновременно использоваться либо в секциях входа и выхода, либо в критических и некритических секциях. При этом хоть чтение данной переменной в цикле и является рабочим алгоритмом, но для практических задач он не подходит из-за забивания канала память–процессор операциями чтения для проверок. Одним из подходов для решения проблемы забивания канала являются алгоритмы локальной прокрутки (local-spin), где процесс при входе в критическую секцию проверяет локальные копии некоторой переменной (или переменных) вместо проверки публичной переменной.

Первыми алгоритмами локальной прокрутки были алгоритмы прокрутки очереди на основе примитива “чтение–модификация–запись” [8, 9]. В них каждый процесс при входе в критическую секцию записывает свою “позицию” в очереди в виде указателя и ждет, пока ее предшественник не выйдет из секции выхода. В случае системы с общим кэшем они требуют максимум O(1) операций над памятью, но в случае когда у процессов нет общего кэша, то невозможно установить верхнюю границу числа операций над памятью.

Позднее Янг и Андерсон [10] предложили алгоритм на основе примитива “чтение–запись”, где из процессов строится двоичное дерево. В качестве листов дерева лежат сами процессы, а в узлах хранится информация, кто должен исполнять критическую секцию из пары потомков. И когда процесс входит в критическую секцию, он пытается захватить контроль над родительским узлом, а когда выходит из секции, то отпускает его. При этом данный алгоритм на любой системе требует всего лишь O(log N) процессов.

Следующим подклассом решений являются “быстрые” алгоритмы исключения. В них существует путь исполнения за константное время для случая, когда всего один процесс хочет

исполнять критическую секцию. Например, алгоритм за авторством Лампорта [11], который тоже строит дерево, но при помощи “разделителя”, который гарантирует, что не более одного процесса может остановиться в конкретном узле, а другие будут в потомках узла. Сам концепт разделителя в отрыве от самого алгоритма оказался настолько популярным, что его используют для уменьшения пространства имен другие алгоритмы [12, 13, 14]. Без соревнования данный алгоритм требует всего семь операций, но при соревновании данный алгоритм не имеет верхней границы на число операций.

Дальнейшие подклассы алгоритмов с исключением в основном сфокусированы на том, чтобы как можно быстрее решить, какие процессы исключают друг друга, но при этом оставить как можно быстрее путь в случае отсутствия конфликта. Для этого используется как статический анализ – прямая модификация алгоритма Лампорта с локальной прокруткой [15], так и динамический – алгоритм Стиера [16] (напрямую использующий Лампорта), Чоу и Сингха [17] (строящий при помощи “заполнителя” несбалансированное дерево со сбалансированными потомками в левом потомке), анализирующие соревновательность в точке истории и на некотором интервале истории [12].

До этого считалось, что один процессор исполняет один процесс одновременно. В реальности процессор может переключаться между различными процессами по истечению отведенных квантов времени. Алгоритм Фишера [18] требует четкого знания о длительности кванта времени для процесса. Идея состоит в том, что в переменной для исключения хранится номер процесса в критической секции. Когда процесс пытается зайти в секцию (в переменной 0), то при захвате секции он пишет туда свой номер и ждет квант. И если в переменной остался его номер, то можно заходить. Алгоритм Алюра, Аттия и Таунберфелда [19] ослабляет требование точного знания кванта до знания верхней границы оценки кванта. При этом априорное знание данной границы не требуется для алгоритма, для уточнения текущей оценки границы используется алгоритм Лампорта.

Теперь перейдем к подклассу алгоритмов, где не требуется наличия атомарных операций. К сожалению, число таких алгоритмов достаточно мало. Первая причина – нет нужды разрабатывать данные алгоритмы. Поскольку во всех существующих популярных процессорах есть поддержка на аппаратном уровне атомарности примитива “чтение–запись”, то алгоритмы без него интересны только с научной точки зрения. Второй причиной является сложность данных подходов. Отсутствие атомарных операций часто требует очень аккуратной работы с переменными, отведенными под общение между процессами, вдобавок к необходимости правильной разметки критических секций.

Из значимых работ можно отметить серию из двух работ за авторством Лампорта [20, 21], где при попытке входа в критическую секцию процесс опрашивает другие процессы об их состоянии. Сложность процесса опроса зависит от того, насколько строгое требование по честности захода в критическую секцию.

Из главных особенностей подхода Лампорта можно отметить поднятие вопроса об устойчивости к ошибкам и постановке четырех критериев устойчивости:

- Безопасность при выключении – схема не ломается, если процесс завершил свою работу, потому что процесс выставил все переменные для общения в базовые значения и тогда только завершился.
- Безопасность при прерывании – схема не ломается, если процесс прервали в критической секции, потому что процесс выставил все переменные для общения в базовые значения и перешел в некритическую секцию (и далее он может продолжать свою работу).
- Безопасность при ошибках – схема не ломается, если процесс встретил ошибку при своей работе, потому что процесс просто постепенно мягко выключится.
- Самостабилизация – если процесс вошел в ошибочное состояние и далее ошибок не будет, то он вернется в нормальное состояние со временем.

Алгоритмы разрешения конфликтов при помощи сообщений. Технически схема Лампорта из [20, 21] может быть и отнесена сюда, поскольку он уже использовал сообщения для обеспечения общей памяти. И в основном большая часть схем взаимного исключения использует сообщения именно с такой целью – превратить частные переменные в подобие публичных переменных для хранения состояния о секции. Именно поэтому они редко когда показывают сильно отличающиеся результаты [22, 23]. С одной стороны, схемы с общей памятью обычно четко фиксируют число процессов, когда исключение при помощи сообщений может быть организовано среди переменного числа процессов [22]. Но с другой стороны, при автоматизации промышленных производств такие ситуации редки, поэтому дальше взаимное исключение при помощи сообщений рассматриваться отдельно от исключения при помощи разделяемой памяти не будет.

Алгоритмы без взаимного исключения. Основная проблема, с которой сталкиваются разработчики схем без взаимного исключения процессов, – отсутствие поддержки необходимых атомарных примитивов в системах команд современных процессоров. Поэтому данные алгоритмы включают реализацию требуемых атомарных примитивов на основе уже существующих, которая проигрывает реализациям алгоритмов со взаимным исключением по скорости.

Например, в работе [24] авторы смогли подобрать набор инструкций для эффективной реализации требуемых атомарных примитивов. Однако авторы рассматривают ситуации, в которых работа алгоритма приводит к рассинхронизации между различными процессами. Авторы предлагают решать эту проблему через проверку корректности чтения, что занимает дополнительное время.

Критический анализ существующих подходов

Среди всех алгоритмов разрешения конфликтов при параллельных вычислениях в контексте автоматизации производств можно отметить несколько общих проблем. Одна из основных проблем состоит в том, что в основном схемы имеют сложность O(log N) по числу операций. При этом стоит отметить, что производственные процессы состоят из множества мелких операций и, следовательно, программа для автоматизации данных производств будет склонна к большому числу процессов внутри себя. Следовательно, стоимость решения возможных конфликтов может быстро расти, даже если не каждый процесс будет конфликтовать за конкретный ресурс.

Дополнительно стоит отметить, что для разрешения конфликтов требуется организовывать межпроцессную коммуникацию, что требует либо памяти, либо канала между процессами. Другим следствием сильной ограниченности ресурсов является организация многопоточности на малом числе физических потоков. И в случае использования квантов времени для организации такого рода многопоточности может возникнуть проблема, когда много процессов просто ожидают один, хотя при другом распределении квантов или при другом размере процессы могли бы просто работать друг за другом. Если бы планировщик не использовал кванты времени, то данной проблемы можно было и избежать, но тогда возникает требование знания верхней границы времени работы каждого процесса (очень сильное требование).

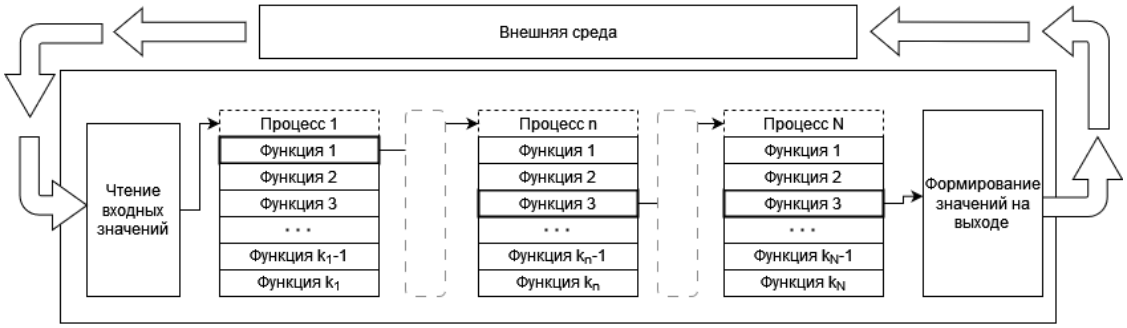
При этом вопрос об отказоустойчивости внутри критической секции перекладывается на разработчика системы полностью. Программист сам должен заботиться о том, чтобы процессы выходили из критической секции нормально в любой ситуации.

Ситуация усложняется в случае мультипроцессорных систем. Даже в варианте системы на одном чипе (SoC) организация межпроцессного взаимодействия усложняется на порядок из-за необходимости организации межпроцессорного взаимодействия и почти гарантированного отсутствия общего кэша. Хотя идея использования мультипроцессорных систем со специализированными под некоторый набор подзадачами является привлекательной для целей автоматизации производства.

Процесс-ориентированная парадигма. Определения и модель

Перед обсуждением, как проблема конкурирующих процессов выглядит в процесс-ориентированном программировании, требуется определить несколько ключевых понятий и описать модель программы.

Процесс-ориентированная парадигма была разработана в ИАиЭ СО РАН [2, 3, 4]. Данная парадигма и реализующие их языки направлены на автоматизацию предприятий и фабрик. Программа в ней представляет собой один гиперпроцесс – упорядоченный набор процессов. Сам же процесс – модифицированный конечный автомат, где у каждого состояния есть некоторая связанная с ней функция. У любого процесса существуют особые состояния: покоя и ошибки. В состоянии покоя процесс может перейти из любого состояния и в нем процесс ничего не делает. И только из состояния покоя процесс может перейти в начальное состояние по внешней причине. Состояние ошибки тоже существует у любого процесса, и оно сигнализирует о произошедшей ошибке в процессе исполнения функции, и из него процесс самостоятельно выйти не может. Функция, связанная с текущим состоянием, исполняется при передаче контроля процессу, и она же отвечает за выбор следующего состояния процесса. Сами функции представляют последовательный набор операций, аналогично процессу из обычных программ. Переменные в процесс-ориентированных программах полностью аналогичны обычным программам. Но у процесса есть важная публичная переменная, в которой он хранит свое текущее состояние.



Цикл процесс-ориентированной программы
Process-oriented program cycle

Исполнение процесс-ориентированной программы происходит в цикле, состоящим из трех частей:

1. Чтение данных из внешней среды.
2. Передача контроля процессам в строго заданном порядке.
3. Запись реакции программы во внешнюю среду.

Таким образом, процесс-ориентированная программа реализует корпоративный параллелизм на одном потоке вычисления. Стоит отметить, что порядок передачи контроля процессам связан с порядковым номером процесса и известен до начала исполнения программы. А сам цикл активируется с некоторым фиксированным заранее известным периодом, и гарантируется, что цикл завершится до начала следующего цикла.

В теории можно ввести состояние функции по аналогии с состоянием процесса, но смысла в этом нет в рамках данной статьи, поскольку далее будет рассматриваться только случай, когда процесс возвращает контроль сам для передачи следующему процессу. Тогда любая функция в процесс-ориентированной программе будет являться атомарным примитивом. Назовем конфликтующими процессами такое подмножество процессов, которые имеют некоторое общее

подмножество публичных переменных, с которыми они работают в рамках одного цикла программы. Конфликт – ситуация, когда от порядка работы конфликтующих процессов зависит состояние программы в конце цикла.

Проблема конфликтов в процесс-ориентированной парадигме

Для удобства анализа рассмотрим некоторую систему, состоящую только из одного клапана. У нее будет всего лишь две операции – открыть и закрыть клапан. Поскольку речь идет об автоматизации производств, то наше вычислительное устройство напрямую работает с содержимым клапана, поэтому любая операция состоит из подачи питания на требуемые элементы в установленном порядке, контроля за состоянием клапана и отключением питания с данных элементов.

Попробуем представить программу в процесс-ориентированной парадигме. Стоит сразу отметить, что единого процесса, скорее всего, не будет. На это есть несколько причин:

- Сама парадигма побуждает дробить алгоритмы на как можно более мелкие элементы. При выполнении данного условия из-за корпоративного параллелизма можно приблизиться к истинному параллелизму (как на ПЛИС). Дополнительно при дроблении на как можно более мелкие элементы отпадает строгая нужда в планировщике и квантовании времени, поскольку квант времени будет порядка времени выполнения одной функции в среднем.
- Процесс является расширением конечного автомата, поэтому он имеет всего одно начальное состояние, что в случае клапана с двумя различными желаемыми результатами будет требовать дополнительное время на обработку “что же хочет пользователь”, что приведет к задержке между сигналом и реакцией системы…

Следовательно, будет два процесса, каждый из которых будет ответствен за свою функцию у клапана. Каждый процесс будет последовательно выполнять функции выставления нужных значений в переменных клапана, анализа состояния клапана и выставления некоторых нейтральных значений в переменные клапана.

Сразу виден сценарий, когда может произойти конфликт – а что если при закрытии клапана мы попробуем открыть клапан? А что делать, если есть ситуации, когда надо закрыть открывающийся клапан? А что если система теперь состоит из нескольких клапанов и есть некоторый порядок, в котором можно открывать и закрывать?

Рассмотрим ситуации по очереди. Первый случай решается относительно просто – надо ввести некоторую публичную переменную, где бы отображалось текущее состояние клапана… И неожиданно такая переменная уже есть – состояние процесса. Поэтому процесс закрытия клапана может проверить состояние открытия клапана и поменять свое поведение. Аналогично решается и вторая проблема – поскольку каждый процесс имеет доступное для всех состояние, то процесс открытия может в начале функции контроля состояния клапана просто проверять состояние процесса закрытия клапана и узнать, что совершается аварийное закрытие клапана. Поскольку набор процессов известен заранее, то данная проверка будет являться проверкой переменной из кэша, а не некоторой процедурой опроса другого процесса.

Перед рассмотрением последнего вопроса рассмотрим несколько иной. Допустим, теперь в системе есть монитор, где отображается текущее состояние клапана. Естественно, его заполняет некоторый отдельный процесс и возникает вопрос, конфликтует ли данный процесс с другими двумя. Формально да, поскольку результат на мониторе будет зависеть от момента, когда процесс отображения будет выполняться в цикле относительно двух других процессов. Но при этом задержка состояния на мониторе и реального состояния системы будет все время стабильной и предсказуемой. Тем более если программист желает разрешить данный конфликт, то из-за гарантии строго порядка исполнения процессов ему достаточно просто раз-

местить процесс отображения после процессов управления путем выставления большего порядкового номера процесса.

Вернемся к последовательной активации нескольких клапанов. Учитывая предыдущий конфликт и его решение, имеется два варианта разрешения конфликта:

1. Всего один процесс на открытие всей системы и один на закрытие.
2. Следить за состоянием некоторых клапанов и не позволять открываться до открытия предыдущих по цепочке. Данный вариант позволяет легко реализовать сложные и разветвленные схемы зависимостей и заодно гарантировать начало открытия клапанов в рамках одного цикла.

Как видно, в большинстве случаев конфликты в рамках процесс-ориентированной программы решаются либо правильной расстановкой порядковых номеров процессов, либо использованием состояний процессов как примитивов синхронизации.

Преимущества и недостатки использования известных методов решения конфликтов в процесс-ориентированных программах

Несмотря на то что методы решения конфликтов в процесс-ориентированных и обычных программах схожи между собой из-за использования идеи критической секции, все же есть некоторые различия между ними и получаемым результатом при разрешении конфликтов. Поэтому можно говорить о преимуществах и недостатках процесс-ориентированных программ в данной сфере по сравнению с обычными программами.

Сразу стоит отметить отсутствие операций с неопределенным временем исполнения (ввод-вывод как пример). Данный факт позволяет обойтись без прерывания хода исполнения процессов из-за истечения некоторого выделенного кванта времени. Поэтому программист будет четко знать места, где программа может потерять управление, и разрабатывать процесс как череду четко определенных критических секций. По итогу каждый отдельный процесс будет захватывать контроль над секцией каждый раз, что может позволить избежать проблемы истощения и взаимной блокировки. Стоит отметить, что в таком случае только на процессе лежит ответственность за реализацию честных вычислений, и если некоторая функция будет иметь неоправданно долгое время исполнения (или же бесконечное), то вся программа может остановиться.

Вторым важным элементом процесс-ориентированной программы является существование явного приоритета всех процессов относительно друг друга и гарантия соблюдения данного приоритета в процессе исполнения. В предыдущем разделе было показано, как данный приоритет позволяет избежать необходимости реализации любого метода решения конфликтов путем правильной расстановки порядкового номера. Таким образом, несмотря на то что как бы конфликт и есть формально, но дополнительных вычислительных усилий он не требует.

Существование состояний у процессов является как положительной стороной процесс-ориентированной программы, так и отрицательной. С одной стороны, их можно использовать в других процессах для разрешения конфликтов и реализации взаимного исключения через состояния процессов. С другой стороны, состояния всех процессов придется все равно синхронизировать между всеми процессам, и в случаях систем с раздельным кэшем это может повлечь большие проблемы, особенно учитывая, что в одной процесс-ориентированной программе число процессов можно оценить цифрой 1000.

Но тут есть пара интересных моментов. Хоть явно и существует требование о том, что любой процесс знает про состояния всех других процессов, но на практике произвольному процессу не интересна большая часть процессов. Если построить граф, где вершинами будут являться процессы, а ребрами будут показаны зависимости процессов от состояний других процессов, то в большинстве случаев такой граф можно поделить на подграфы, которые будут либо являться компонентами связности в исходном графе, либо будут иметь очень малое число

ребер с другими подграфами (относительно ребер внутри). Тогда каждый подграф можно будет исполнять на своем вычислительном устройстве. Данное разбиение уменьшит сложность разрешения конфликтов не только из-за минимизации общения между вычислительными устройствами, а еще потому, что можно будет решать данную задачу в рамках одного подграфа, который будет иметь порядок, сравнимый с числом процессов в обычной программе.

Дополнительно стоит отметить наличие состояния ошибки, в которое достаточно легко перейти и явно сигнализирующее об ошибке в процессе исполнения. При этом поскольку переход в данное состояние аналогичен переходу в любое другое состояние, и если процесс реализует безопасность при выключении, то безопасность при прерывании будет тоже обеспечена. Поскольку вся функция и является критической секцией, то безопасность при ошибке аналогична безопасности при прерывании. Следовательно, для реализации безопасных параллельных вычислений достаточно иметь безопасность при завершении работы функции и самовосстановление. Стоит отметить, что из состояний покоя и ошибки можно перейти только в начальное состояние, где и будут содержаться инициализаторы для локальных переменных. Следовательно, безопасность при завершении работы уже реализована в процесс-ориентированной парадигме и от разработчика требуется только реализовать некоторые процедуры по восстановлению из ошибок, запускающиеся по переходу процесса или процессов в состояние ошибки.

Заключение

В статье была рассмотрена проблема конфликтов при параллельном исполнении различных процессов в программе в контексте промышленной автоматизации. Данный контекст отличается не только требованием реализации данного исполнения на маломощных вычислительных устройствах, но также требованием отказоустойчивости. К сожалению, уже существующие подходы хоть и могут удовлетворять первому требованию, но вопрос отказоустойчивости остается слабо изученным. Поэтому в статье была рассмотрена процесс-ориентированная парадигма как способ написания программ, удовлетворяющим требованиям.

В результате анализа было показано, что процесс-ориентированная программа позволяет решать проблему конфликтов при параллельном исполнении различных процессов путем решения проблемы конкурирующих процессов в данной программе. При этом использование процесс-ориентированной парадигмы позволяет избежать излишних затрат на критические секции и проблем с неправильным определением данных секций разработчиком программы. Дополнительно сама парадигма из-за отсутствия необходимости в дополнительных примитивах синхронизации не требует ввода специальных атомарных операций со стороны производителей вычислительных устройств. Для разработчиков при использовании парадигмы основное преимущество заключается в автоматическом выполнении трех из четырех требований для организации отказоустойчивой параллельной программы.

В качестве дальнейшей работы в этом направлении предлагается рассмотреть вопрос о необходимости требования разделения программы на как можно мелкие процессы. При рассмотрении схемы из нескольких клапанов было сделано предположение, что каждым клапаном управляет свой процесс. Но что, если это не так? Что, если у процесса могло бы быть несколько начальных состояний? Ну и, естественно, остается открытым вопрос о возможности создания средств обнаружения конфликтов в процесс-ориентированной программе при статическом анализе кода.

Список литературы

1. **John K. H., Tiegelkamp M.** Programming Industrial Automation Systems // IEC 61131-3. 2010. DOI: 10.1007/978-3-642-12015-2

2. **Зюбин В. Е.** Язык Рефлекс. Математическая модель алгоритмов управления // Датчики и системы. 2006. № 5. С. 24–30.
3. **Зюбин В. Е.** Статическая балансировка вычислительных ресурсов в процесс-ориентированном программировании // Вестник НГУ. Серия: Информационные технологии. 2012. Т. 10. Вып. 2. С. 44–54.
4. **Зюбин В. Е. и др.** Базовый модуль, управляющий установкой для выращивания монокристаллов кремния //Датчики и системы. 2004. №. 12. С. 17–22.
5. **Булавский Д. В. и др.** Автоматизированная система управления установкой для выращивания монокристаллов кремния //Автометрия. 1996. №. 2. С. 26.
6. **Dijkstra E. W.** Solution of a Problem in Concurrent Programming Control, Pioneers and Their Contributions to Software Engineering, p. 289–294, 2001, DOI: 10.1007/978-3-642-48354-7_10
7. **Knuth D. E.** Additional comments on a problem in concurrent programming control, Communications of the ACM, vol. 9, no. 5, p. 321–322, May 1966, DOI: 10.1145/355592.365595
8. **Anderson T. E.** The performance of spin lock alternatives for shared-money multiprocessors, IEEE Transactions on Parallel and Distributed Systems, vol. 1, no. 1, p. 6–16, 1990, DOI: 10.1109/71.80120
9. **Mellor-Crummey J. M., Scott M. L.** Algorithms for scalable synchronization on shared-memory multiprocessors, ACM Transactions on Computer Systems, vol. 9, no. 1, p. 21–65, Feb. 1991, DOI: 10.1145/103727.103729
10. **Kim Y.-J., Anderson J. H.** A space- and time-efficient local-spin spin lock, Information Processing Letters, vol. 84, no. 1, p. 47–55, Oct. 2002, DOI: 10.1016/s0020-0190(02)00224-7
11. **Lamport L.** A fast mutual exclusion algorithm, ACM Transactions on Computer Systems, vol. 5, no. 1, p. 1–11, Jan. 1987, DOI: 10.1145/7351.7352
12. **Afek Y., Attiya H., Fouren A., Stupp G., Touitou D.** Long-lived renaming made adaptive, Proceedings of the eighteenth annual ACM symposium on Principles of distributed computing, May 1999, DOI: 10.1145/301308.301335
13. **Afek Y., Merritt M.** Fast, wait-free (2k-1)-renaming, Proceedings of the eighteenth annual ACM symposium on Principles of distributed computing, May 1999, DOI: 10.1145/301308.301338
14. **Attiya H., Fouren, A.** Adaptive long-lived renaming with read and write operations (No. CS Technion report CS0956), Computer Science Department, Technion, 1999.
15. **Michael M. M., Scott M. L.** Fast Mutual Exclusion, Even with Contention, Jun. 1993, DOI: 10.21236/ada272947
16. **Styer E.** Improving fast mutual exclusion, Proceedings of the eleventh annual ACM symposium on Principles of distributed computing - PODC ’92, 1992, DOI: 10.1145/135419.135453
17. **Choy M., Singh A. K.** Adaptive solutions to the mutual exclusion problem, Distributed Computing, vol. 8, no. 1, p. 1–17, Aug. 1994, DOI: 10.1007/bf02283567.
18. **Alur R., Attiya H., Taubenfeld G.** Time-adaptive algorithms for synchronization, Proceedings of the twenty-sixth annual ACM symposium on Theory of computing - STOC ’94, 1994, DOI: 10.1145/195058.195464
19. **Kuhn F., Maus Y., Weidner S.** Deterministic Distributed Ruling Sets of Line Graphs, Lecture Notes in Computer Science, p. 193–208, 2018, DOI: 10.1007/978-3-030-01325-7_19
20. **Lamport L.** The mutual exclusion problem: part I - A theory of interprocess communication, Journal of the ACM, vol. 33, no. 2, p. 313–326, Apr. 1986, DOI: 10.1145/5383.5384
21. **Lamport L.** The mutual exclusion problem: Part II - Statement and solutions, Journal of the ACM, vol. 33, no. 2, p. 327–348, Apr. 1986, DOI: 10.1145/5383.5385
22. **Schneider J., Elkin M., Wattenhofer R.** Symmetry breaking depending on the chromatic number or the neighborhood growth, Theoretical Computer Science, vol. 509, p. 40–50, Oct. 2013, DOI: 10.1016/j.tcs.2012.09.004

23. **Daymude J. J., Richa A. W., Scheideler C.** Local mutual exclusion for dynamic, anonymous, bounded memory message passing systems, arXiv.org, 24-Feb-2022. [Online]. URL: https://arxiv.org/abs/2111.09449 (accessed on: 19.02.2023)
24. **Fraser K., Harris T.** Concurrent programming without locks, ACM Transactions on Computer Systems, vol. 25, no. 2, p. 5, 2007

References

1. **John K. H., Tiegelkamp M.** Programming Industrial Automation Systems // IEC 61131-3. 2010. DOI: 10.1007/978-3-642-12015-2
2. **Zyubin V. E.** Reflex Language. Mathematical model of control algorithms // Sensors and Systems. 2006. Vol. 5. Pp. 24–30. (in Russ.)
3. **Zyubin V. E.** Static balancing of computing resources in process-oriented programming // Vestnik NSU. Series: Information Technologies. 2012. Vol. 10, no. 2. Pp. 44–54. (in Russ.)
4. **Zyubin V. E. et al.** Basic module controlling the installation for growing single crystals of silicon // Sensors and Systems. 2004. Vol. 12. Pp. 17–22. (in Russ.)
5. **Bulavskiy D. V. et al.** Automated control system for a facility for growing single crystals of silicon // Autometric. 1996. Vol. 2. P. 26. (in Russ.)
6. **Dijkstra E. W.** Solution of a Problem in Concurrent Programming Control // Pioneers and Their Contributions to Software Engineering. 2001. Pp. 289–294. DOI 10.1007/978-3-642-48354-7_10
7. **Knuth D. E.** Additional comments on a problem in concurrent programming control // Communications of the ACM. 1966. Vol. 9, no. 5. Pp. 321–322. DOI 10.1145/355592.365595
8. **Anderson T. E.** The performance of spin lock alternatives for shared-money multiprocessors // IEEE Transactions on Parallel and Distributed Systems. 1990. Vol. 1, no. 1. Pp. 6–16. DOI 10.1109/71.80120
9. **Mellor-Crummey J. M., Scott M. L.** Algorithms for scalable synchronization on shared-memory multiprocessors // ACM Transactions on Computer Systems. 1991. Vol. 9, no. 1. Pp. 21–65. DOI 10.1145/103727.103729
10. **Kim Y.-J., Anderson J. H.** A space- and time-efficient local-spin spin lock // Information Processing Letters. 2002. Vol. 84, no. 1. Pp. 47–55. DOI 10.1016/s0020-0190(02)00224-7
11. **Lamport L.** A fast mutual exclusion algorithm // ACM Transactions on Computer Systems. 1987. Vol. 5, no. 1. Pp. 1–11. DOI 10.1145/7351.7352
12. **Afek Y., Attiya H., Fouren A., Stupp G., Touitou D.** Long-lived renaming made adaptive // Proceedings of the eighteenth annual ACM symposium on Principles of distributed computing. 1999. DOI 10.1145/301308.301335
13. **Afek Y., Merritt M.** Fast, wait-free (2k-1)-renaming // Proceedings of the eighteenth annual ACM symposium on Principles of distributed computing. 1999. DOI 10.1145/301308.301338
14. **Attiya H., Fouren A.** Adaptive long-lived renaming with read and write operations (No. CS Technion report CS0956). Computer Science Department, Technion, 1999.
15. **Michael M. M., Scott M. L.** Fast Mutual Exclusion // Even with Contention. 1993. DOI 10.21236/ada272947
16. **Styer E.** Improving fast mutual exclusion, Proceedings of the eleventh annual ACM symposium on Principles of distributed computing. PODC'92, 1992. DOI 10.1145/135419.135453
17. **Choy M., Singh A. K.** Adaptive solutions to the mutual exclusion problem // Distributed Computing. 1994. Vol. 8, no. 1. Pp. 1–17. DOI 10.1007/bf02283567
18. **Alur R., Attiya H., Taubenfeld G.** Time-adaptive algorithms for synchronization // Proceedings of the twenty-sixth annual ACM symposium on Theory of computing. STOC'94, 1994. DOI 10.1145/195058.195464

19. **Kuhn F., Maus Y., Weidner S.** Deterministic Distributed Ruling Sets of Line Graphs // Lecture Notes in Computer Science. 2018. Pp. 193–208. DOI 10.1007/978-3-030-01325-7_19
20. **Lamport L.** The mutual exclusion problem: part I – A theory of interprocess communication // Journal of the ACM. 1986. Vol. 33, no. 2. Pp. 313–326. DOI 10.1145/5383.5384
21. **Lamport L.** The mutual exclusion problem: Part II – Statement and solutions // Journal of the ACM. 1986. Vol. 33, no. 2. Pp. 327–348. DOI 10.1145/5383.5385
22. **Schneider J., Elkin M., Wattenhofer R.** Symmetry breaking depending on the chromatic number or the neighborhood growth // Theoretical Computer Science. 2013. Vol. 509. P. 40–50. DOI 10.1016/j.tcs.2012.09.004
23. **Daymude J. J., Richa A. W., Scheideler C.** Local mutual exclusion for dynamic, anonymous, bounded memory message passing systems [Online]. URL: https://arxiv.org/abs/2111.09449 (accessed on: 19.02.2023).
24. **Fraser K., Harris T.** Concurrent programming without locks // ACM Transactions on Computer Systems. 2007. Vol. 25, no. 2. P. 5.

Информация об авторе

Пермяшкин Дмитрий Андреевич, старший инженер ООО «Техкомпания Хуавэй»

Information about the Author

Dmitry A. Permyashkin, Senior Engineer of Huawei Tech Company LLC, Moscow, Russian Federation

Статья поступила в редакцию 20.03.2023;
одобрена после рецензирования 01.06.2023; принята к публикации 01.06.2023

The article was submitted 20.03.2023;
approved after reviewing 01.06.2023; accepted for publication 01.06.2023

Научная статья

УДК 528, 004.9
DOI 10.25205/1818-7900-2023-21-2-18-28

Периодизация развития геоинформационных технологий как части информационных технологий

Антон Сергеевич Флеенко

Российский университет транспорта (РУТ – МИИТ)
Москва, Россия

fleenkospb@mail.ru, https://orcid.org/0000-0002-2829-9361

Аннотация

Феномен информатизации общества достаточно хорошо освещен в научной и философской литературе. При этом анализу развития инструмента информатизации общества – информационных технологий – уделено меньше внимания. Одной из актуальных проблем изучения информационных технологий является определение этапов их развития. Применяя исторический подход к изучению информационных технологий (ИТ) и их ответвления – геоинформационных технологий (ГИТ), возможно установить закономерности их развития и определить направления дальнейшего роста. В процессе исследования становления и совершенствования ИТ и ГИТ выделен ряд подходов к подразделению истории развития данных технологий на этапы. Для первого определяющим фактором перехода к следующему этапу развития является совершенствование средств обработки информации. В рамках данного подхода выделено четыре основных этапа – «элементарный», «механический», «программный» и «сетевой». Второй подход связан с эволюцией решаемых ИТ и ГИТ задач – рассмотрен переход от формализуемых к частично формализуемым и неформализуемым вопросам. Третий основан на анализе развития ИТ и ГИТ как инновационных технологий, последовательно проходящих стадии инвенции, диффузии и адаптации. В процессе анализа истории ИТ установлено, что во второй половине XX века произошло выделение ГИТ как одного из средств решения частично формализуемых и неформализуемых задач, что привело к началу «явного» развития ГИТ как самостоятельного направления ИТ.

Ключевые слова

геоинформационные системы, информационная технология, новая информационная технология, инновация, геоинформатика

Для цитирования

Флеенко А. С. Периодизация развития геоинформационных технологий как части информационных технологий // Вестник НГУ. Серия: Информационные технологии. 2023. Т. 21, № 2. С. 18–28. DOI 10.25205/1818-7900-2023-21-2-18-28

Development of Geographic Information Systems as an Information Technology

Anton S. Fleenko

Russian University of Transport (MIIT)
Moscow, Russian Federation

fleenkospb@mail.ru, https://orcid.org/0000-0002-2829-9361

Abstract

The phenomenon of informatization of society is quite well covered in the scientific and philosophical literature. At the same time, less attention is paid to the analysis of the development of the tool for informatization of society — information technology. One of the urgent problems of studying information technologies is to determine the stages of their development. Applying a historical approach to the study of information technologies (IT) and their branches — geoinformation technologies (GIT) — it is possible to establish the patterns of their development and determine the directions for further growth. In the process of studying the formation and improvement of IT and GIT, approaches to dividing the history of the development of these technologies into stages are determined. For the first approach, the determining factor in the transition to the next stage of development is the improvement of information processing tools. Within the framework of this approach, four main stages are distinguished - "elementary", "mechanical", "software" and "network". The second approach is related to the evolution of the tasks solved by IT and GIT — the transition from formalized to partially formalized and non-formalized issues is considered. The third approach is based on the analysis of IT development and GIT as innovative technologies that successively go through the stages of invention, diffusion and adaptation. In the process of analyzing the history of IT, it was found that after the middle of the 20th century, GIT was singled out as one of the means of solving partially formalized and non-formalized tasks, which led to the beginning of the "explicit" development of GIT as an independent direction of IT.

Keywords

geoinformation systems, information technology, new information technology, innovation, geoinformatics

For citation

Fleenko A. S. Development of Geographic Information Systems as an Information Technology. *Vestnik NSU. Series: Information Technologies*, 2023, vol. 21, no. 2, pp. 18–28. DOI 10.25205/1818-7900-2023-21-2-18-28

Введение

Эволюция технологий значительно опережает их осмысление научным сообществом. Новые идеи появляются практически ежедневно, развитие часто происходит спонтанно и скачкообразно. Изучение закономерностей возникновения и распространения информационных технологий, становления информации в качестве важной составляющей общественных отношений представляется важной задачей. Представляется возможным в реальном времени проследить эволюцию общества, с иного угла посмотреть на протекавшие в прошлом и протекающие сейчас процессы, выявить новые закономерности и предсказать дальнейшие пути развития.

Одним из основополагающих методов в естественных науках является моделирование, что обусловлено сложностью исследуемых объектов, явлений и процессов. К примеру, модели земной поверхности с расположенными на них природными и антропогенными объектами являются продуктом таких наук, как геодезия и картография. Развитие информационных технологий и рост информатизации общества приводит к качественному изменению моделирования и возникновению новых направлений научной деятельности. Так, закономерности построения цифровых пространственных моделей в географических информационных системах становятся предметом изучения относительно молодой науки геоинформатики.

Значительную роль в изучении проблемы информатизации сыграли исследователи развития общества и общественных отношений второй половины XX века. Среди них – американский философ и социолог Дэниел Белл, один из основоположников концепции постинду-

стриального общества [1], американский философ и социолог Элвин Тоффлер, описывающий процесс перехода к такому типу общества и акцентирующий внимание на значении информации в нем [2, 3]. Информатизации и автоматизации посвящены работы советских ученых и общественных деятелей Анатолия Ивановича Китова и Акселя Ивановича Берга, заложивших основы применения информационных технологий в различных сферах хозяйственной деятельности в нашей стране в послевоенный период [4, 5, 6, 7].

К концу XX – началу XXI века как ответ на все большее расширение границ информатизации растет количество исследований, связанных с данной темой. Большой вклад в концепцию информационного общества внес российский ученый и философ Анатолий Ильич Ракитов [8], рассматривающий проблему информатизации с точки зрения философии и методологии науки. Академик Аркадий Дмитриевич Урсул [9] акцентирует внимание на связи процессов информатизации и устойчивого развития, формирования ноосферы, тем самым развивая идеи Владимира Ивановича Вернадского о новом шаге в развитии человеческого общества. Доктор философских наук Игорь Серафимович Мелюхин систематизирует отечественный и зарубежный опыт в области информатизации, критически оценивает его в существующих реалиях [10, 11]. Ученый и изобретатель Рифгат Фаизович Абдеев проводит философское осмысление произошедших в результате информатизации перемен в обществе [12]. Канадский ученый Дон Тапскотт посвятил множество работ изучению экономических процессов в информационном обществе [13], кроме того, значительный объем исследований в области информатизации связан с образованием [14], медициной, правовыми и социальными проблемами [15] информационного общества.

Таким образом, проблема информатизации широко освещена в научной литературе. Перечисленные работы направлены в основном на изучение теоретических основ информационного общества и процессов информатизации. Подробно рассмотрено значение информационных технологий в различных сферах хозяйственной деятельности. При этом в условиях продолжающегося развития общества и роста объемов информации недостаточно изучена значимость формирования, развития и внедрения инструментов работы с ней – информационных технологий.

Рассматривая в качестве предмета исследования геоинформационные системы как ответвление информационных технологий, можно определить основные этапы становления технологий работы с пространственными данными и направления их дальнейшего развития. В связи с этим заявляемая в данной статье цель научной работы, обозначаемая как установление периодизации развития геоинформационных технологий с учетом их связи с информационными технологиями в целом, представляется актуальной.

Методика

Для достижения поставленной цели использовался ряд методов: анализ научной литературы, сбор данных, их систематизация и классификация, исторический метод, а также группа методов системного анализа.

– *Аналитический метод* выступает в качестве основного метода исследования. Он связан с анализом сложных объектов и процессов, в рамках которого они разделяются на части и изучаются по отдельности. В рамках анализа собранные в процессе исследования научной литературы данные и факты были систематизированы, выделены основные закономерности, на основе которых построены классификации.

– Использование *системного анализа* как совокупности методов, направленных на познание сложных объектов посредством представления их в виде взаимосвязанных между собой элементов, направлено на установление закономерностей и взаимосвязей между элементами – их систематизацию. Это помогает решить ряд проблем, таких как слабая структурированность исходных данных, неопределенность, многоаспектность и комплексность информации [16].

Метод классификации при этом используется в качестве вспомогательного с целью установления порядка в исследуемых объектах и явлениях с помощью разделения их по группам на основе определенных правил [17].

– *Исторический метод*, основанный на принципе историзма, рассматривает объект исследования через развитие. Целостность и взаимосвязанность прошлого, настоящего и будущего объектов и процессов становятся основой для их изучения [18]. В данном контексте рассмотрена научная литература по теме работы.

Геоинформационные технологии как новая информационная технология

Определяющей чертой современного общества является провозглашение информации как одного из главных ресурсов. Необходимость работы с крупными массивами данных диктуют рост глобализационных процессов, усложнение объектов научного исследования, популярность концепции устойчивого развития. Вследствие этого информационные технологии (далее – ИТ) как совокупность инструментов, средств и процессов работы с информацией пронизывают все сферы деятельности человека.

Согласно А. А. Ручкову [14], информационная технология представляет собой совокупность методов и технических средств сбора, организации, хранения, обработки, передачи и предоставления информации, расширяющих знания людей и развивающих их возможности по управлению техническими и социальными процессами. В научных исследованиях часто под новыми информационными технологиями понимаются информационные технологии, использующие компьютерные средства [19].

Геоинформационная система (далее – ГИС) является основной технологией сбора, обработки, анализа и выдачи пространственной информации, а также средством ее интерактивного моделирования и получения новой [20]. Таким образом, ГИС выступает в качестве ИТ, центральным объектом которой является пространственная информация, или геоданные.

Для стабильного существования человеческого общества необходимо постоянное совершенствование инструментов работы с информацией. Согласно проведенному анализу литературы, периодизация развития ИТ условна и зависит от выбора определяющего фактора, изменение которого приводит к переходу к другому периоду. Рассмотрим подробнее наиболее распространенные факторы, среди которых:

– технические средства работы с информацией;
– характер решаемых задач;
– инновационность внедряемых технологий.

Подход к периодизации развития ИТ и ГИТ на основе этапов развития технических средств обработки информации

В развитии вычислительной техники как инструмента преобразования исходной информации в результат посредством вычисления [21] выделено четыре основных этапа.

Особенность первого, «элементарного», этапа – отсутствие самих средств вычислительной техники в современном понимании. Начало развитию информационных технологий положило появление информации – тех данных, которые получал и использовал человек в своей деятельности с самого зарождения общества и культуры. На данном этапе формируются средства обмена информацией – естественные и формальные языки. Истоки систематизации и классификации данных восходят к трудам античных философов. Информация передавалась преимущественно в устной форме, позже – письменно. Изобретение в XV веке печатного станка стало количественным скачком в передаче информации, однако не определило переход к следующему этапу, поскольку не связано с качественными изменениями в обработке данных. Таким образом, начальный этап, длившийся до XVII века, связан с возникновением и совер-

шенствованием элементов информационных технологий: происходит накопление и систематизация данных, развитие языков, способов обработки и представления информации.

Для геоинформационных технологий первый этап связан с созданием и обработкой первой пространственной информации – в виде точек изображались звезды на ночном небе, а также ориентиры на местности проживания. Первые карты, основанные на расчетах, появились в Вавилоне в XV–XIV веках до нашей эры. Дальнейшее совершенствование обработки и интерпретации пространственных данных привело к созданию картографических проекций. Изобретение компаса и астрономических приборов позволило получать более точную пространственную информацию, а возникновение типографий – создавать более точные карты и планы местности. Таким образом, на данном этапе происходит развитие картографии как фундамента для ГИС-технологий.

Начало второго, «механического», этапа (XVII – середина XX века) обусловлено развитием точных наук, повлекшим за собой разработку механизмов, выполнявших простейшие арифметические операции. Например, суммирующая машина Блеза Паскаля (середина XVII века) и механический арифмометр Готфрида Лейбница (конец XVII века) были созданы с целью упрощения работы с математическими операциями. Английский математик Чарльз Бэббидж в середине XIX века проводит масштабные теоретические и практические исследования по разработке вычислительных машин, исключающих многочисленные ошибки в сложных расчетах. Отличительной особенностью второго этапа является качественный скачок в информационных технологиях – теоретически обоснована и практически доказана возможность создания сложных вычислительных машин.

Итогом второго этапа развития для геоинформационных технологий стало становление «прародительницы» геоинформатики – картографической науки, определение ее основных инструментов, методов и средств. ГИТ развиваются в контексте картографии, согласно периодизации развития инструментария для измерений и съемок на местности по А. М. Берлянту в это время создаются первые средства дистанционного зондирования Земли (вторая половина XIX века) [22], использование которых приводит к росту количества и качества геоданных.

Третий этап (середина – конец XX века) развития информационных технологий – «программный» – характеризуется устранением важного недостатка вычислительных машин прошлого – созданием эффективных систем ввода и вывода данных. В конце XIX века создается табулятор с перфокартами, которые расшифровываются электрическим током и поступают в вычислительную машину. Задачи прогнозирования погодных условий, расшифровки закодированных сообщений, необходимость которых диктовали масштабные военные действия XX века, были решены с использованием вычислительных машин. Окончание Второй мировой войны стало отправной вехой для эры электронно-вычислительных машин (далее – ЭВМ). В настоящее время историки информатики выделяют четыре поколения ЭВМ, с каждым из которых скорость внесения информации и обработки растет, программный набор усложняется, а сама вычислительная техника становится все более компактной. Третий этап развития характеризуется совершенствованием путей ввода, обработки, хранения и передачи информации, превращением информации в ресурс, созданием и совершенствованием языков программирования.

Как следствие роста количества источников геоданных возникает потребность в их обработке – период с середины XX века для геоинформационных технологий является началом их становления и активного развития. Появляется векторная графика, первые программные продукты, связанные с обработкой картографических изображений (перевод из растрового формата в векторный, автоматическая оцифровка). Так, согласно А. М. Берлянту [22], в этот период в картографической науке возникают цифровые и электронные методы и технологии составления карт, появляется геоинформационное картографирование.

Четвертый, «сетевой», этап (конец XX века – настоящее время) в истории информационных технологий происходит в условиях технологической революции, связанной с инфор-

матизацией, цифровизацией и компьютеризацией общества. Информация становится главной ценностью, а скорость работы с ней, эффективность ее хранения и передачи играют значительную роль во всех сферах жизни. Начало современного этапа развития опирается на два процесса – организацию всемирной системы связанных между собой компьютеров и персонализацию использования компьютеров, их повсеместное распространение.

В рамках четвертого этапа развития информационных технологий геоинформатика становится самостоятельной наукой, происходит широкое распространение программного обеспечения, которое влечет за собой рост профессиональных и непрофессиональных пользователей геоинформационных систем. С этого времени картографическая наука развивается отдельно, но в тесной связи с геоинформатикой. Сейчас ГИС-технологии используются во многих сферах деятельности, а информация, полученная в геоинформационных системах, становится основой для принятий решений в градостроительной и иных сферах деятельности.

Подход к периодизации ИТ и ГИТ на основе эволюции решаемых задач

Первоначально информационные технологии служили средством автоматизации формализуемых операций, для которых известны все элементы и внутренние взаимосвязи. Информационные системы прежде всего были инструментом математики, физики и других точных наук, финансовой и торговой деятельности, повышая эффективность расчетов и уменьшая количество ошибок. Те же особенности характерны и для пространственных задач, решаемых в рамках картографии, – различные проекции и уточнение методов и приборов повышали точность и эффективность использования карт. Одним из примеров решения формализуемых задач в геоинформационных системах является перевод координат из одной координатной системы в другую, осуществляемый на основе строгих математических параметров.

На втором этапе развития информационных технологий операции перестали быть детерминированными, обрели в своем составе стохастическую компоненту. Необходимость решения частично формализуемых или неформализуемых задач обусловила совершенствование использования информационных технологий. Теперь их использование позволяет строить модели объектов и явлений, прогнозировать их поведение во времени, проводить глубокий анализ данных и получать новые в его процессе для более эффективного принятия решений.

Создание картографических материалов с применением ИТ положило основу для цифрового пространственного моделирования с использованием совокупности математических, статистических и картографических приемов. Таким образом, создание первых геоинформационных систем относится к началу использования информационных технологий для решения неформализуемых задач и отчасти является следствием этого процесса. К примеру, в качестве частично формализуемой задачи может являться анализ растровых изображений с помощью инструментария геоинформационных систем и построение на основе данного анализа тематических карт.

Подход к периодизации развития ИТ и ГИТ как инновационного продукта

Информационные технологии (как и геоинформационные) по множеству критериев, среди которых новизна, системность, направленность на высокую результативность [23], можно отнести к инновациям. Жизненный цикл инновации по Йозефу Шумпетеру [24] включает в себя стадии инвенции, диффузии и адаптации. Согласно проведенному анализу, эволюция информационных и геоинформационных технологий полностью отвечает процессу развития глобального инновационного продукта, однако инновационные стадии для информационных технологий и геоинформационных технологий уже не соответствуют друг другу.

Возвращаясь к первому из рассмотренных подходов периодизации развития информационных технологий, первый и второй этапы развития вычислительной техники можно предста-

вить в качестве стадии инвенции ИТ, когда разрабатываются основные научные концепции и изобретательские стратегии. Для стадии инвенции высока значимость личностей-инноваторов, активная деятельность которых связана с собственным желанием повысить эффективность работы. Среди таких личностей для ИТ – французский математик Блез Паскаль, немецкий математик Готфрид Лейбниц, английский математик Чарльз Бэббидж.

Третий этап эволюции вычислительной техники совпадает со стадией диффузии ИТ, в рамках которой этот инновационный продукт распространяется. Базой инноваций третьего этапа являются коллективы ученых, при этом роль личности-инноватора отходит на второй план, так как необходим планомерный и постоянный рост технологий. Личности-инноваторы остаются движущей силой подобных групп, например, английский математик Алан Тьюринг или американский математик Джон фон Нейман.

Четвертому этапу развития вычислительных средств соответствует стадия адаптации, в процессе которой инновационный продукт создает условия для трансформации общества, становится доступным и очень распространенным. Роль инноватора претерпевает ряд изменений – рыночная экономика определяет создание крупных корпораций и конкурентной среды, в которой лидеры выступают не только в качестве «генераторов идей», но и как популяризаторы своего продукта – таковы американский бизнесмен Илон Маск, российский разработчик Павел Дуров, американский разработчик Билл Гейтс.

Инновационные стадии можно применить и к периодизации развития геоинформационных технологий, однако заметен временной сдвиг по сравнению с информационными технологиями. Он связан с тем, что начало первым разработкам и пониманию возможностей использования технических средств обработки пространственной информации (стадия инвенции) было положено только в середине XX века, после начала «явного» развития ГИС-технологий. До этого времени происходило развитие картографической науки и пограничных ей областей. Стадия диффузии по отношению к ГИС-технологиям связана с государственной поддержкой развития геоинформационных систем в конце XX века, а коммерческое развитие и появление широкого круга пользователей ГИС-технологий знаменует начало стадии адаптации с начала XXI века. Среди личностей-инноваторов ГИТ в начале «явного» развития геоинформационных систем в условиях стадии адаптации ИТ стоит выделить основателей ведущих компаний по разработке программного обеспечения ГИС – Джек Данджермонд и «ESRI», Шон О'Салливан и «MapInfo Corporation», Гари Шерман и «QGIS».

Выводы

В ходе исследования рассмотрены три основных подхода к разделению истории развития информационных технологий на этапы (см. табл. и рис.):

– первый подход связан с анализом развития средств вычислительной техники;

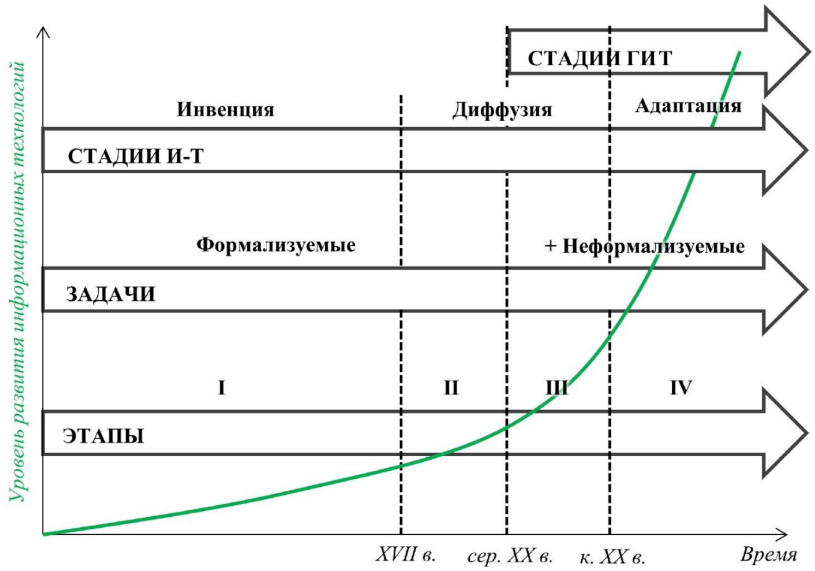
– второй подход основан на анализе эволюции типов задач, решаемых информационными технологиями;

– третий подход рассматривает информационные технологии в качестве инновационного процесса.

Подходы к разделению развития информационных технологий на этапы

Approaches to the Division of Information Technology Development into Stages

Период	Краткая характеристика	Вычислительная техника	Типы задач	Инновационный продукт
до XVII в.	Разработка и совершенствование теорий, связанных с информацией, появление «ручных» средств работы с информацией	1 этап («элементарный»)	Этап формализуемых задач	Стадия инвенции
XVII – сер. XX в.	Совершенствование теорий, связанных с информацией, разработка прототипов «механических» средств работы с информацией	2 этап («механический»)		
сер. – к. XX в.	Формирование и развитие отрасли работы с информацией, становление рынка информационных технологий	3 этап («программный»)	Этап неформализуемых задач	Стадия диффузии
к. XX в. – наше время	Широкое распространение информационных технологий, трансформация отношений «человек–человек», «человек–компьютер»	4 этап («сетевой»)		Стадия адаптации



Графическая интерпретация этапов развития информационных технологий и геоинформационных технологий
Graphical interpretation of the development of information technology and geoinformation technology

Основываясь на выявленных закономерностях развития информационных технологий, можно утверждать, что во 2-й половине XX века, с выделением геионформационных технологий в качестве самостоятельного набора средств решения неформализуемых задач начинается «явное» развитие ГИС-технологий. Предполагается, что их дальнейшее совершенствование будет связано с разработкой и внедрением новых технологий моделирования, использующих новейшие достижения автоматизации, искусственного интеллекта и систем дополненной реальности, совершенствованием системного и математико-картографического анализа для развития использования геоинформационных инструментов [22]. В дальнейшем уже ГИС-технологии могут стать базой для появления совершенно новых инструментов работы с информацией на стыке разных практических и теоретических направлений деятельности.

Список литературы

1. **Белл Д.** Грядущее постиндустриальное общество. Опыт социального прогнозирования / Пер. с англ. 2-е изд., испр. и доп. М.: Academia, 2004. 788 с.
2. **Тоффлер Э.** Третья волна / Пер. с англ. М.: АСТ, 2004. 261 с.
3. **Тоффлер Э.** Шок будущего / Пер. с англ. М.: АСТ, 2002. 557 с.
4. **Воробьев Е. И., Китов А. И.** Введение в медицинскую кибернетику. М.: Медицина, 1977. 287 с.
5. **Китов А. И.** Программирование экономических и управленческих задач. М.: Советское радио, 1971. 371 с.
6. **Китов А. И.** Электронные цифровые машины. М.: Советское радио, 1956. 276 с.
7. **Берг А. И.** О возможностях автоматизации управления народным хозяйством / Проблемы кибернетики: сб. ст. М.: Физматгиз, 1961. Вып. 6. С. 88–100
8. **Ракитов А. И.** Философия компьютерной революции. М.: Политиздат, 1991. 287 с.
9. **Урсул А. Д.** Информатизация общества и переход к устойчивому развитию цивилизации / Вестник Российского общества информатики и вычислительной техники. 1993. Вып. 1–2. С. 35–45.
10. **Мелюхин И. С.** Информационное общество и баланс интересов государства и личности / Информационное общество. 1997. Вып. 4–6. С. 3–26.
11. **Мелюхин И. С.** Концепция «Информационного общества» и кризис / Информационное общество. 1998. Вып. 6. С. 20–22.
12. **Абдеев Р. Ф.** Философия информационной цивилизации. М.: ВЛАДОС, 1994. 336 с.
13. **Тапскотт Дон.** Электронно-цифровое общество: Плюсы и минусы эпохи сетевого интеллекта / Пер. с англ. М.: РЕФЛ-бук, 1999. 403 с.
14. **Ручков А. А.** Информационные технологии в современной системе образования // Вестник Пензенского государственного университета. 2015. № 1 (9). С. 57–60.
15. **Тимошенко Т. В.** Адаптация человека к современной информационной среде // Известия Южного федерального университета. Технические науки. 2012. № 11 (136). С. 155–159.
16. **Новосельцев В. И.** Системный анализ: современные концепции. 2-е изд, испр. и доп. Воронеж: Кварта, 2003. 360 с.
17. **Понкин И. В.** Классификация как метод научного исследования, в частности в юридической науке // Вестник Пермского университета. Юридические науки. 2017. № 37. С. 249–259.
18. **Арзамаскин Ю. Н.** Принцип историзма в научном исследовании // Армия и общество. 2011. №3 (27). С. 7–11.
19. **Минькович Т. В.** Информационные технологии: понятийно-терминологический аспект // Образовательные технологии и общество (ОТО). 2012. № 2 (15). С. 371–389.
20. **Блиновская Я. Ю. Задоя Д. С.** Геоинформационные системы в техносферной безопасности: Учеб. пособие. М.: ИНФРА-М, 2021. 160 с.

21. **Хасанов И. И.** Зарождение информационно-вычислительных систем: основные этапы развития вычислительной техники // История и педагогика естествознания. 2017. № 3. С. 31–36.
22. **Берлянт А. М.** Картография: Учебник для вузов. М.: Аспект Пресс, 2002. 336 с.
23. **Грасмик К. И.** Инновации: сущность, виды, особенности управления // Качество. Инновации. Образование. 2008. № 2 (33). С. 27–35.
24. **Веретенникова А. Ю. Паникарова С. В.** Жизненный цикл социальных инноваций в общественном секторе // Вестник Удмуртского университета. Серия: Экономика и право. 2015. № 6. С. 118–121.

References

1. **Bell D.** The Coming of Post-Industrial Society: A Venture in Social Forecasting: Translation from English / 2nd ed., add. and upd. Moscow: Academia, 2004. p. 788. (in Russ.)
2. **Toffler A.** Third wave: Translation from English. Moscow: AST, 2004. p. 261. (in Russ.)
3. **Toffler A.** Future Shock: Translation from English. Moscow: AST, 2002. p. 557. (in Russ.)
4. **Vorobev E. I., Kitov A. I.** Introduction to medical cybernetics. Moscow: Medicina, 1977. p. 287. (in Russ.)
5. **Kitov A. I.** Programming of economic and managerial tasks. Moscow: Sovetskoe radio, 1971. p. 371. (in Russ.)
6. **Kitov A. I.** Electronic digital machines. Moscow: Sovetskoe radio, 1956. p. 276. (in Russ.)
7. **Berg A. I.** On the possibilities of automating the management of the national economy // Problems of Cybernetics: Digest of scientific articles. 1961. Vol. 6. Pp. 88–100. (in Russ.)
8. **Rakitov A. I.** Philosophy of the computer revolution. Moscow: Politizdat, 1991. p. 287.
9. **Ursul A. D.** Informatization of society and the transition to sustainable development of civilization // The Bulletin of the Russian Society of Informatics and Computer Engineering. 1993. Vol. 1–2. Pp. 35–45. (in Russ.)
10. **Melyuhin I. S.** Information society and the balance of interests of the state and the individual // Information Society journal. 1997. Vol. 4–6. Pp. 3–26. (in Russ.)
11. **Melyuhin I. S.** The concept of the “Information Society” and the crisis // Information Society journal. 1998. Vol. 6. Pp. 20–22. (in Russ.)
12. **Abdeev R. F.** Philosophy of information civilization. Moscow: VLADOS, 1994. p. 336. (in Russ.)
13. **Tapskott D.** Digital society: pros and cons of the era of network intelligence: Translation from English. Moscow: REFL-buk, 1999. P. 403.
14. **Ruchkov A. A.** Information technologies in the modern education system // Vestnik of Penza state university. 2015. Issue 1(9). Pp. 57–60. (in Russ.)
15. **Timoshenko T. V.** Adaptation of the person to the modern information environment // Izvestiya SFedU. Engineering sciences. 2012. Iss. 11(136). Pp. 155–159. (in Russ.)
16. **Novoselcev V. I.** Systems Analysis: Modern Concepts: 2nd ed., add. and update. Voronezh: Kvatra, 2003. p. 360.
17. **Ponkin I. V.** Classification as a method of scientific research, particularly in jurisprudence // Perm University Herald. Juridical sciences. 2017. Iss. 37. Pp. 249–259. (in Russ.). DOI: 10.17072/1995-4190-2017-37-249-259
18. **Arzamaskin Y. N.** The principle of historicism in scientific research // Army and society. 2011. Iss. 3(27). Pp. 7–11. (in Russ.)
19. **Minkovich T. V.** Information technologies: conceptual and terminological aspect // Educational technologies and society. 2012. Iss. 2(15). Pp. 371–389. (in Russ.)
20. **Blinovskaya Y. Y., Zadoya D. S.** Geoinformation systems in technosphere safety: textbook. Moscow: INFRA-M, 2021. p. 160. (in Russ.)

21. **Khasanov I. I.** The emergence of information systems: the main stages of computer development // History and pedagogy of natural science. 2017. Iss. 3. Pp. 31–36. (in Russ.)
22. **Berlyant A. M.** Cartography: Textbook for universities. Moscow: Aspekt Press, 2002. p. 336. (in Russ)
23. **Grasmik K. I.** Innovations: essence, types, features of management // Quality. Innovation. Education. 2008. Iss. 2(33). Pp. 27–35. (in Russ.)
24. **Veretennikova A. Y., Panikarova S. V.** Life cycle of social innovations in public sector // Bulletin of Udmurt University. Series History and Philology. 2015. Iss. 6. Pp. 118–121. (in Russ.)

Информация об авторах

Флеенко Антон Сергеевич, аспирант кафедры геодезии, геоинформатики и навигации Института пути, строительства и сооружений Российского университета транспорта (РУТ – МИИТ)

Information about the Author

Anton S. Fleenko, postgraduate student of the Department of Geodesy, Geoinformatics and Navigation of the Institute of Railway Track, Construction and Structures of the Russian University of Transport (MIIT)

Статья поступила в редакцию 14.02.2023;
одобрена после рецензирования 12.05.2023; принята к публикации 12.05.2023

The article was submitted 14.02.2023;
approved after reviewing 12.05.2023; accepted for publication 12.05.2023

Научная статья

УДК 004.421.2: 81’33: 519.178
DOI 10.25205/1818-7900-2023-21-2-29-38

Использование платформы ТХМ корпусного анализа для анализа текстов сообществ социальных сетей

Алина Игоревна Фокина[1],
Александр Андреевич Чеповский[2],
Андрей Михайлович Чеповский[3]

[1–3]Национальный исследовательский университет «Высшая школа экономики» (НИУ ВШЭ)
Москва, Россия

[3]Российский университет дружбы народов (РУДН)
Москва, Россия

[1]aifokina@edu.hse.ru
[2]aachepovsky@hse.ru
[3]chepovskiy-am@rudn.ru.

Аннотация

При формировании графов взаимодействующих объектов, построенных при импорте данных из социальных сетей и сетей мгновенного обмена сообщениями, в качестве атрибутов вершин выступают в том числе и текстовые данные. В настоящей работе авторы приводят описание методики исследования текстов, основанной на процедурах корпусного анализа. Целью данной статьи является проверка методологических средств, предоставляемых программным обеспечением ТХМ для сравнительного анализа текстов выделенных сообществ на графе взаимодействующих объектов. Метод предлагается для оценки качества выделения неявных сообществ на графе, полученном при импорте данных из сети каналов мессенджера Telegram.

Ключевые слова

анализ социальных сетей, автоматический анализ текстов, платформа ТХМ

Для цитирования

Фокина А. И., Чеповский А. А., Чеповский А. М. Использование платформы ТХМ корпусного анализа для анализа текстов сообществ социальных сетей // Вестник НГУ. Серия: Информационные технологии. 2023. Т. 21, № 2. С. 29–38. DOI 10.25205/1818-7900-2023-21-2-29-38

Using TXM Platform of Corpus Analysis for Text Analysis of Social Media

Alina I. Fokina[1], Aleksandr A. Chepovskiy[2], Andrey M. Chepovskiy[3]

[1–3]HSE University
Moscow, Russian Federation

[3]Federal Research Center "Informatics and Management"
Moscow, Russian Federation

[1]aifokina@edu.hse.ru
[2]aachepovsky@hse.ru
[3]chepovskiy-am@rudn.ru.

Abstract

When forming graphs of interacting objects built when importing data from social networks and instant messaging networks, text data also act as vertex attributes. In this paper, the authors describe a text research methodology based on corpus analysis procedures. The purpose of this article is to test the methodological tools provided by the TXM software for the comparative analysis of the revealed communities texts on the graph of interacting objects. The method is proposed to assess the quality of the implicit communities revealing on the graph obtained by importing data from the channel network of the Telegram messenger.

Keywords

social network analysis, automated text analysis, TXM platform

For citation

Fokina A. I., Chepovskiy A. A., Chepovskiy A. M. Using TXM Platform of Corpus Analysis for Text Analysis of Social Media. *Vestnik NSU. Series: Information Technologies*, 2023, vol. 21, no. 2, pp. 29–38. DOI 10.25205/1818-7900-2023-21-2-29-38

Введение

Изучение структуры сетевых сообществ актуально для анализа социальных связей и сетей мгновенного обмена сообщениями в контексте средств распространения информации, что имеет большое значение в современном обществе. Исследование возникающих в таких сетях сообществ позволяет определять процессы распространения информации, выделения криминальных групп, корректировать использование каналов маркетинговых коммуникаций [9, 11, 13].

Для решения задач выделения сетевых сообществ пользователей в работе [5] был разработан метод Галактик, который в процессе своего применения обеспечивает выделение пересекающихся неявных сообществ. Под выделением неявных сообществ на графе подразумевается разбиение графа на подграфы такое, что плотность связей внутри этих подграфов намного выше плотности связей между ними. При этом существенно выделение на графе пересекающихся сообществ, подразумевающих наличие общих вершин, принадлежащих сразу двум или более сообществам. Именно такие сообщества и выделяются методом, разработанным в [5].

При выделении сообществ одной из наиболее сложных проблем является оценка корректности и эффективности работы соответствующих методов [9, 14]. Поэтому проблема оценки качества выделения сообществ на графах ставится как актуальный вопрос информационных технологий [11].

В работах [1, 8] было показано, что для оценки качества исследования сетей Telegram-каналов удобным инструментом является сочетание алгоритмического подхода по выделению сообществ в совокупности с анализом лингвистических характеристик. Это позволяет выделять группы каналов, ведущих активное информационное воздействие, подтверждать корректность

выделения неявных сообществ. В [1, 8] была показана возможность применять сравнение психолингвистических факторов и методику сравнения частотных словарей текстов для подтверждения корректности выделения неявных сообществ в графах, полученных при импорте данных из сетей мгновенного обмена сообщениями.

В данной работе для сравнительного анализа текстов сообществ, выделяемых на графе Telegram-каналов, мы предлагаем использовать методы корпусного анализа на базе платформы TXM [12], которые развивались в [2, 3, 4, 7, 13] для выявления дифференцирующих признаков текстов различной природы.

Формирование корпусов текстов

Исследования проводились на импортированных данных из Telegram-каналов. Импорт данных и формирование графа взаимодействующих объектов осуществлялись согласно (U, M, R)-модели информационного взаимодействия, описывающей распространение информации за импортируемый промежуток времени [6], в которой учитывается количество внешних ссылок в постах (U), количество постов (M) и количество репостов (R) с весовыми коэффициентами.

Первый граф G_1 был получен скачиванием данных посредством обхода графа, начиная с канала @kudago, соответствующего развлекательному сайту http://kudago.com/msk/. При импорте данных для этого графа выбирался временной интервал с 03.10.2022 по 17.10.2022. Глубина обхода графа бралась равной 5. Исходный скаченный граф G_1 состоит из 619 вершин и 2973 ребер.

Второй граф G_2 был получен скачиванием данных посредством обхода графа, начиная с канала @ob_obraz, который посвящен новостям общего и высшего образования РФ. За временной интервал был взят период с 01.12.2022 по 31.01.2023. Глубина обхода графа бралась равной 2. Исходный скаченный граф G_2 состоит из 168 вершин и 697 ребер.

Третий граф G_3 был получен скачиванием данных посредством обхода графа, начиная с нескольких каналов, освещающих ход специальной военной операции Российской Федерации. За временной интервал был взят период с 01.07.2022 по 01.09.2022. Глубина обхода графа бралась равной всего лишь равной 1, ибо в противном случае подключалось много крупных политических каналов, что несколько искажало исследуемую картину. Исходный скаченный граф G_3 состоит из 600 вершин и 18 009 ребер.

К полученным графам G_1, G_2 и G_3 был применен метод Галактик [5] и получены разбиения на неявные пересекающиеся сообщества. При обработке графа G_1 в процессе работы алгоритма некоторые вершины и инцидентные им ребра убираются из графа как не участвующие активно во взаимодействии, поэтому после работы метода у графа G_1 осталось 458 вершин. Эти вершины распределились по 8 выделенным пересекающимся сообществам. При обработке графа G_2 аналогичным методом осталось 89 изначальных вершин, из которых были сформированы так же 8 сообществ. По результатам обработки графа G_3 было получено 288 вершин, на которых выделено 18 сообществ. Таким образом, для каждого из трех графов сформированы сообщества $Community_i$, где $i=0, \ldots, 7$ для графов G_1 и G_2 и $i=0, \ldots, 17$ для графа G_3.

Для каждого из сообществ $Community_i$ были скачаны текстовые сообщения всех каналов – членов этих сообществ за исследуемый период. Все тексты каждого сообщества объединялись в единый для сообщества массив текстов на естественном языке. В таблице приведены размеры этих массивов текстов. Из текстов удалялись специальные имена (имена аккаунтов, почтовые адреса) и далее рассматривались массивы текстов на естественном языке. Полученные массивы текстов сообществ рассматривались как подкорпуса корпуса текстов каждого из трех графов. Поэтому в таблице приведены размеры полученных подкорпусов, измеренные в количестве русскоязычных словоупотреблений как основной единицы корпусного анализа.

Размеры корпусов текстов сообществ

Community Text Corpus Sizes

Сообщество	Корпуса текстов графа G_1		Корпуса текстов графа G_2		Корпуса текстов графа G_3	
	Размер (Кб)	Число словоупотреблений	Размер (Кб)	Число словоупотреблений	Размер (Кб)	Число словоупотреблений
$Community_0$	335	29293	988	35120	3498	309125
$Community_1$	39208	3450208	985	30942	1806	179034
$Community_2$	13571	1201393	566	20239	3633	303847
$Community_3$	3485	302679	782	26759	1169	100055
$Community_4$	1633	138912	1021	39702	2975	260058
$Community_5$	37897	3272627	8914	322660	750	67982
$Community_6$	2925	244706	11881	437942	5730	446270
Community7	10794	952898	1981	69321	703	59385
$Community_8$	–	–	–	–	4134	422629
$Community_9$	–	–	–	–	11489	463161
$Community_{10}$	–	–	–	–	3659	308019
$Community_{11}$	–	–	–	–	9298	441104
$Community_{12}$	–	–	–	–	11350	452082
$Community_{13}$	–	–	–	–	5398	463593
$Community_{14}$	–	–	–	–	71538	455354
$Community_{15}$	–	–	–	–	17063	443049
$Community_{16}$	–	–	–	–	4243	358384
$Community_{17}$	–	–	–	–	6660	448108

Методы лингвистического анализа

Сформированные, как описано выше, массивы текстов сообществ исследовались методами корпусного анализа на базе платформы TXM [12]. Платформа TXM является эффективным средством, позволяющим проводить комплексный анализ корпусов текстов процедурами анализа соответствий, кластеризации, построения лексических таблиц, поиска сложных лексических конструкций.

В текстах выделялись словоупотребления, для которых проводился автоматический морфологический анализ словоформ. В настоящей работе используется программный пакет TreeTagger [15], предоставляющий возможность совместного морфологического анализа слов предложения на основе статистической модели. По результатам работы TreeTagger определяются канонические (начальные) формы слова. Преимуществом пакета является однозначность морфологического анализа словоупотреблений.

Наборы текстов с вычисленными характеристиками импортируются в пакет TXM для последующего анализа. В рамках платформы выделяются подкорпуса, представляющие собой объединенный текст публикаций членов каждого выделенного в графе неявного сообщества.

Для исследований корпусов текстов сообществ мы используем анализ соответствий [10], который включен как один из инструментов в платформу TXM. Анализ соответствий является методом исследования корпуса текстов, который разделен на подкорпуса. При делении текстов

на подкорпуса есть возможность интерпретировать близость между значениями характеристик подкорпусов как оценку, указывающую на сходство или различие между этими подкорпусами.

Данный метод состоит в анализе частот совместного появления значений переменных в таблице частот лингвистических характеристик. В основе этого лежит изучение симметричной матрицы, сопоставляющей признаки друг другу. Такой подход позволяет увидеть взаимосвязи между признаками, а также дает возможность интерпретировать выделенные факторы как совокупность некоторого набора выделенных из текстов характеристик.

В процессе решения задачи разделения подкорпусов методом анализа соответствий исследуется частота совместного появления значений определенных переменных, в качестве которых рассматриваются выделенные при формировании корпусов лингвистические характеристики.

Результаты работы метода наглядно представляются в графической интерпретации анализа соответствий (рис. 1–3), которая иллюстрирует пространственное расположение подкорпусов в зависимости от частот совместного появления значений исследуемых переменных (лингвистических характеристик).

Результаты анализа корпусов текстов сообществ

Описанные выше корпуса для графов (см. табл.) были проанализированы с использованием описанной выше функциональности ТХМ «анализ соответствий». Детально были исследованы следующие лексические объекты: словоформы; начальные формы слов с морфологическими характеристиками, полученные с помощью TreeTagger; начальные формы слов, отнесенные к различным частям речи.

На рисунках представлены пространственные расположения подкорпусов на основе слов для подкорпусов текстов сообществ графа G_1 (рис. 1), на основе нормальных форм слов для подкорпусов текстов сообществ графа G_2 (рис. 2), на основе существительных для подкорпусов текстов сообществ графа G_3 (рис. 3).

В названии осей указывается процент вариации по корпусу характеристик, входящих в выделенный процедурой анализа соответствий фактор. По осям откладывается характеристика степени отклонения набора признаков от указанной в названии оси процента вариации для конкретного подкорпуса. Таким образом, чем дальше от пересечения осей координат расположен подкорпус по осям координат, тем сильнее в нем отличаются наборы признаков (частота встречаемости) от признаков по корпусу.

На рис. 1 представлены пространственные расположения подкорпусов по результатам анализа соответствий для частотных таблиц слов для корпуса текстов сообществ графа G_1. Точки подкорпусов сообществ $Community_0$ и $Community_2$ лежат во втором квадранте, сообществ $Community_4$ и $Community_5$ лежат в третьем квадранте, сообществ $Community_3$ и $Community_7$ лежат в четвертом квадранте, а $Community_6$ лежит во первом квадранте. На диаграмме подкорпуса разнесены в презентационном пространстве результатов анализа соответствий.

На рис. 2 представлены пространственные расположения подкорпусов по результатам анализа соответствий для частотных таблиц нормальных форм слов для корпуса текстов сообществ графа G_2. Точка подкорпуса сообщества $Community_6$ лежит на оси координаты фактора 1, $Community_5$ лежит в первом квадранте, точки подкорпусов сообществ $Community_1$, $Community_2$, $Community_3$ и $Community_7$ лежат во втором квадранте, сообществ $Community_0$ и $Community_4$ лежат в четвертом квадранте. Таким образом, как и в первом примере, подкорпуса разнесены в презентационном пространстве результатов анализа соответствий.

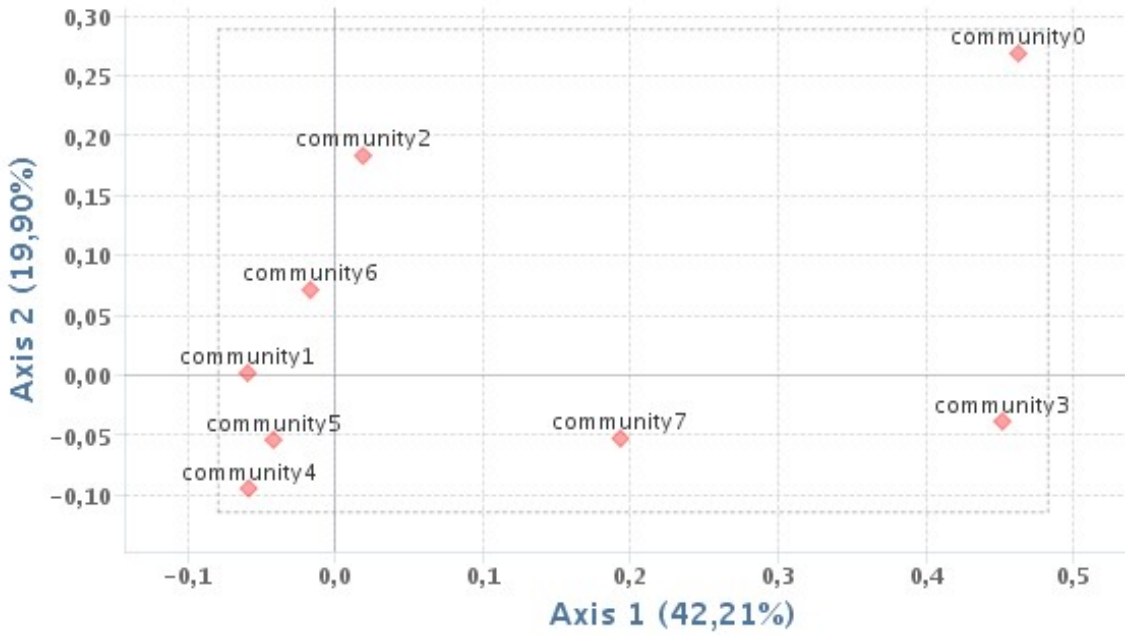


Рис. 1. Анализ соответствий для корпуса сообществ графа G_1 для слов
Fig. 1. Words Correspondence Analysis for the corpus of communities of graph G_1

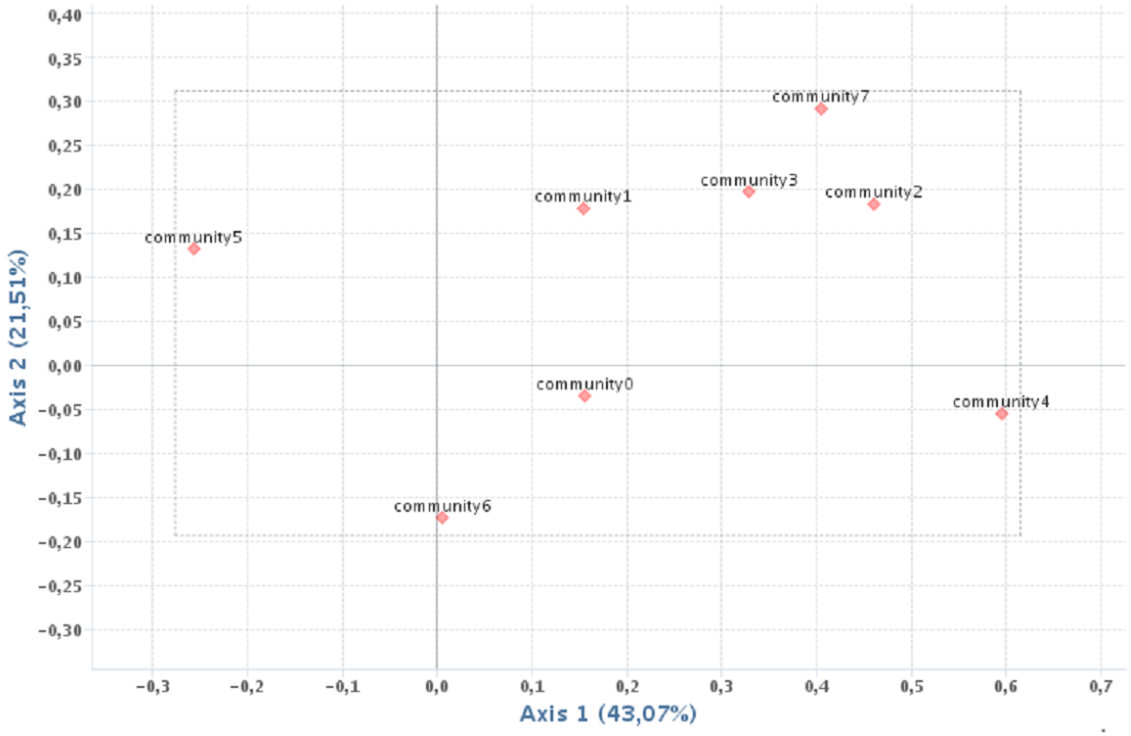


Рис. 2. Анализ соответствий для корпуса сообществ графа G_2 для нормальных форм слов
Fig. 2. Lemmas Correspondence Analysis for the corpus of communities of graph G_2

На рис. 3 представлены пространственные расположения подкорпусов по результатам анализа соответствий для частотных таблиц существительных для корпуса текстов сообществ

графа G_3. Видно, что по аналогии с первым и вторым примерами подкорпуса разнесены в презентационном пространстве результатов анализа соответствий.

Представленные на рис. 1, 2 и 3 результаты анализа соответствий для всех корпусов трех исследуемых графов явно показывают разделение подкорпусов текстов выделенных неявных пересекающихся сообществ по лингвистическим характеристикам. Это, по нашему мнению, можно рассматривать как свойство сообществ графов взаимодействующих объектов, которое может подтверждать корректность выделения сообществ.

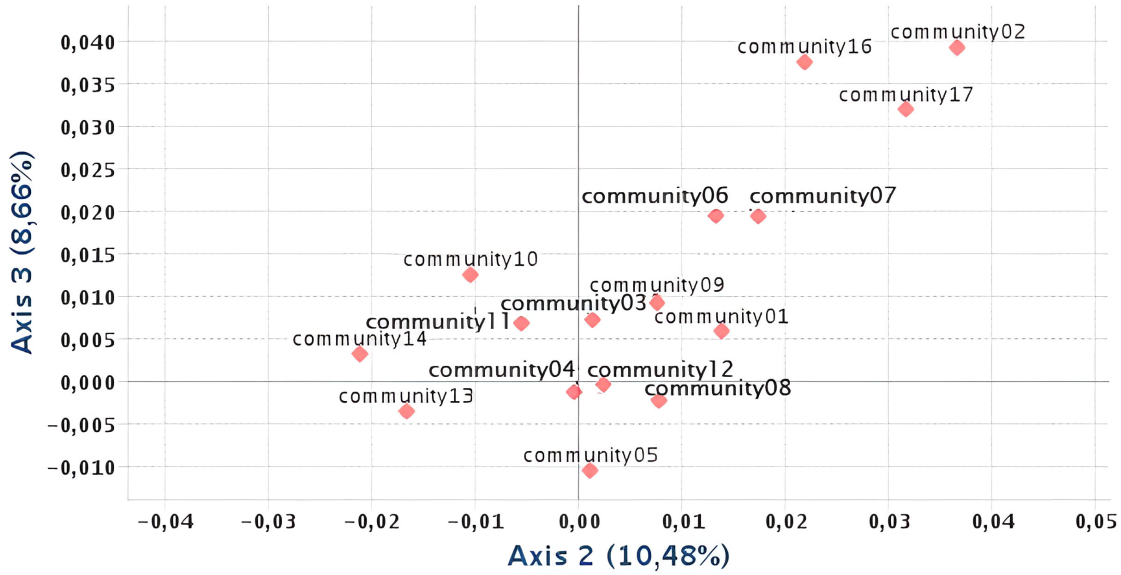


Рис. 3. Анализ соответствий для корпуса сообществ графа G_3 для существительных
Fig. 3. Noun Correspondence Analysis for the corpus of communities of graph G_3

Заключение

Предложена и опробована методика для исследования методами корпусного анализа текстов, сформированных из атрибутивных данных групп вершин графов взаимодействующих объектов. Для различных групп общения проведены исследования текстов выявленных неявных сообществ таких графов, а именно графов информационного взаимодействия сети Telegram-каналов.

В рамках работы к наборам текстов выделенных сообществ были успешно применены средства анализа корпусной платформы TXM. Проведен анализ соответствий для различных лингвистических характеристик корпусов текстов сообществ (слов, начальных форм, частей речи).

На основе исследования реальных данных показана возможность оценки корректности выделения пересекающихся сообществ на графе информационного взаимодействия. Данный подход основан на анализе методами компьютерной лингвистики объединенных корпусов текстов, составленных по публикациям каналов, входящих в выделенные сообщества.

Данная работа вместе с работами [1, 8] формулирует общую комплексную методику оценки корректности выделения пересекающихся сообществ на графах взаимодействующих объектов.

Список литературы

1. **Аванесян Н. Л., Соловьев Ф. Н., Чеповский А. А.** Характеристики текстов сообществ социальных сетей // Вестник НГУ. Серия: Информационные технологии. 2021. Т. 19, № 1. С. 5–14. DOI 10.25205/1818-7900-2021-19-1-5-14.
2. **Лаврентьев А. М., Рябова Д. М., Тихомирова Е. А., Фокина А. И., Чеповский А. М., Шерстинова Т. Ю.** Сравнительный анализ специальных корпусов текстов для задач безопасности // Вопросы кибербезопасности. 2020. № 3(37). С. 58–65. DOI: 10.21681/2311-3456-2020-03-58-65.
3. **Лаврентьев А. М., Смирнов И. В., Соловьев Ф. Н., Суворова М. И., Фокина А. И., Чеповский А. М.** Анализ корпусов текстов террористической и антиправовой направленности // Вопросы кибербезопасности. 2019. № 4(32). С. 54–60. DOI: 10.21681/2311-3456-2019-4-54-60.
4. **Лаврентьев А. М., Соловьев Ф. Н., Суворова М. И., Фокина А. И., Чеповский А. М.** Новый комплекс инструментов автоматической обработки текста для платформы TXM и его апробация на корпусе для анализа экстремистских текстов // Вестник НГУ. Серия: Лингвистика и межкультурная коммуникация. 2018 Т. 16 № 3 С. 19–31. DOI 10.25205/1818-7935-2018-16-3-19-31.
5. **Попов В. А., Чеповский А. А.** Выделение неявных пересекающихся сообществ на графе взаимодействия Telegram-каналов с помощью «метода Галактик» // Труды ИСА РАН. Т. 72. 4/2022. С. 39–50. DOI: 10.14357/20790279220405.
6. **Попов В. А., Чеповский А. А.** Модели импорта данных из мессенджера Telegram // Вестник НГУ. Серия: Информационные технологии. 2022. Т. 20. № 2. С. 60–71. DOI: 10.25205/1818-7900-2022-20-2-60-71.
7. **Соловьев Ф. Н.** Автоматическая обработка текстов на основе платформы TXM с учетом анализа структурных единиц текста // Вестник НГУ. Серия: Информационные технологии. 2020. Т. 18. №1. С. 74–82. DOI 10.25205/1818-7900-2020-18-1-74-82.
8. **Чеповский А. А.** Об особенностях построения и анализа графов взаимодействующих объектов в сети Telegram.-каналов. Вопросы кибербезопасности. 2023; 1(53):75-81. DOI:10.21681/2311-3456-2023-1-75-81.
9. **Чеповский А. А.** О неявных сообществах на графе взаимодействующих объектов. Успехи кибернетики. 2023;4(1):56–64. DOI: 10.51790/2712-9942-2023-4-1-08.
10. **Benzécri J.-P.** L'analyse des données: l'analyse des correspondances. 2nd ed. Paris: Dunod, 1979. Vol. 2.
11. **Fortunato, S., Newman, M. E. J.** 20 years of network community detection. Nat. Phys. 2022; 18:848–850.
12. **Heiden, S.** The TXM Platform: Building Open-Source Textual Analysis Software Compatible with the TEI Encoding Scheme. In: Proceedings of the 24th Pacific Asia Conference on Language, Information and Computation. Sendai, Japan. P. 389–398.
13. **Lavrentiev A., Sherstinova T., Chepovskiy A., Pincemin B.** Using TXM Platform for Research on Language Changes over Time: The Dynamics of Vocabulary and Punctuation in Russian Literary Texts // Vestnik Tomskogo Gosudarstvennogo Universiteta, Filologiya. 2021. Vol. 70. P. 69-89. DOI: 10.17223/19986645/70/5.
14. **Newman M. E. J.** Networks: An Introduction. Oxford University Press, 2010. 784 p.
15. **Schmid H.** Probabilistic Part-of-Speech Tagging Using Decision Trees // Proc. of International Conference on New Methods in Language Processing. Manchester, UK. 1994. URL = http://www.cis.uni-muenchen.de/sschmid/tools/TreeTagger/data/tree-tagger1.pdf. (дата обращения: 30.05.2023).

References

1. **Avanesyan N. L, Solovev F. N., Chepovskiy A. A.** Characteristics of Texts of Social Networks Communities // Vestnik NSU. Series: Information Technologies. 2021. Vol. 19(1). Pp. 5–14. (in Russ.) DOI 10.25205/1818-7900-2021-19-1-5-14
2. **Lavrentiev A. M., Raybova D. M., Tikhomirova E. A., Fokina A. I., Chepovskiy A. M., Sherstinova T. Yu.** Comparative analysis of special text corpora for security-related tasks // Voprosi kiberbezopasnosti. 2020. № 3(37). Pp. 58–65. (in Russ.) DOI 10.21681/2311-3456-2020-03-58-65
3. **Lavrentiev A. M., Smirnov I. V., Solovev F. N., Suvorova M. I., Fokina A. I., Chepovskiy A. M.** Analiz korpusov tekstov terroristicheskoi i antipravovoy napravlennosti // Voprosi kiberbezopasnosti. 2019. № 4(32). Pp. 54–60. (in Russ.) DOI 10.21681/2311-3456-2019-4-54-60
4. **Lavrentiev A. M., Solovev F. N., Suvorova M. I., Fokina A. I., Chepovskiy A. M.** A New Toolkit for Natural Text Processing with the TXM Platform and its Application to a Corpus for Analysis of Texts Propagating Extremist Views // Vestnik NSU. Series: Linguistics and Intercultural Communication. 2018. Vol. 16, № 3. Pp. 19–31. (in Russ.). DOI 10.25205/1818-7935-2018-16-3-19-31
5. **Popov V. A., Chepovskij A. A.** Vydelenie neyavnyh peresekayushchihsya soobshchestv na grafe vzaimodejstviya Telegram-kanalov s pomoshch'yu «metoda Galaktik» // Trudy ISA RAN. 2022. Vol. 72. № 4. Pp. 39–50. (in Russ.). DOI 10.14357/20790279220405
6. **Popov V. A., Chepovskij A. A.** Telegram Messenger Data Import Models // Vestnik NSU. Series: Information Technologies. 2022. Vol. 20, № 2. Pp. 60–71. (in Russ.). DOI 10.25205/1818-7900-2022-20-2-60-71
7. **Solovev F. N.** Embedding Additional Natural Language Processing Tools into the TXM Platform // Vestnik NSU. Series: Information Technologies. 2020. Vol. 18, no. 1. Pp. 74–82. (in Russ.). DOI 10.25205/1818-7900-2020-18-1-74-82
8. **Chepovskiy A. A.** On the construction and analysis of graphs of interacting objects in the Telegram-channels network // Voprosy kiberbezopasnosti. 2023. Vol. 1(53). Pp. 75–81. (in Russ.). DOI 10.21681/2311-3456-2023-1-75-81
9. **Chepovskiy A. A.** Implicit Communities Defined on the Graph for Interacting Objects // Russian Journal of Cybernetics. 2023. Vol. 4(1). Pp. 56–64. (in Russ.). DOI 10.51790/2712-9942-2023-4-1-08
10. **Benzécri J.-P.** L'analyse des données: l'analyse des correspondances. 2nd ed. Paris: Dunod, 1979. Vol. 2.
11. **Fortunato S., Newman M. E. J.** 20 years of network community detection // Nat. Phys. 2022. Vol. 18. Pp. 848–850.
12. **Heiden S.** The TXM Platform: Building Open-Source Textual Analysis Software Compatible with the TEI Encoding Scheme // Proceedings of the 24th Pacific Asia Conference on Language, Information and Computation. Sendai, Japan. Pp. 389–398.
13. **Lavrentiev A., Sherstinova T., Chepovskiy A., Pincemin B.** Using TXM Platform for Research on Language Changes over Time: The Dynamics of Vocabulary and Punctuation in Russian Literary Texts // Vestnik Tomskogo Gosudarstvennogo Universiteta, Filologiya. 2021. Vol. 70. Pp. 69–89. DOI 10.17223/19986645/70/5
14. **Newman M. E. J.** Networks: An Introduction. Oxford University Press, 2010. 784 p.
15. **Schmid H.** Probabilistic Part-of-Speech Tagging Using Decision Trees [Online] // Proc. of International Conference on New Methods in Language Processing. Manchester, UK. 1994. URL: http://www.cis.uni-muenchen.de/sschmid/tools/TreeTagger/data/tree-tagger1.pdf (accessed on: 30.05.2023).

Информация об авторах

Фокина Алина Игоревна, аспирант Национального исследовательского университета «Высшая школа экономики»

Чеповский Александр Андреевич, кандидат физико-математических наук, доцент Национального исследовательского университета «Высшая школа экономики»

Чеповский Андрей Михайлович, доктор технических наук, профессор Российского университета дружбы народов (РУДН)

Information about the Authors

Alina I. Fokina, postgraduate student, National Research University Higher School of Economics Moscow, Russia

Alexander A. Chepovskiy, Ph.D. (mathematics), Associate Professor, National Research University Higher School of Economics Moscow, Russia

Andrey M. Chepovskiy, Dr Sc. (Eng), Peoples Friendship University of Russia (RUDN University)

Статья поступила в редакцию 02.06.2023;
одобрена после рецензирования 27.06.2023; принята к публикации 27.06.2023

The article was submitted 02.06.2023;
approved after reviewing 27.06.2023; accepted for publication 27.06.2023

Научная статья

УДК 004.032.26
DOI 10.25205/1818-7900-2023-21-2-39-50

Применение нейросетевого моделирования в задачах прогнозирования уровня паводка рек

Татьяна Михайловна Шамсутдинова

Башкирский государственный аграрный университет
Уфа, Россия

tsham@rambler.ru, https://orcid.org/0000-0003-1809-3615

Аннотация

Цель данной статьи – рассмотреть теоретические и практические вопросы разработки нейросетевых моделей для прогнозирования паводка рек (на примере реки Белая в районе г. Уфы), а также реализовать соответствующую нейронную сеть на языке Python. Для построения обучающей выборки были использованы архивные данные метеослужб и сайтов метеонаблюдений за паводковые периоды реки Белая (Агидель) 2018–2022 годов. Были собраны и проанализированы следующие показатели: уровень воды, температура воды, дневная и ночная температура воздуха, осадки, высота снежного покрова, включая сведения о предпаводковом состоянии снежного покрова. Программная реализация нейронной сети выполнялась с использованием библиотеки глубокого обучения PyTorch; кроме этого, использовались модули библиотек Matplotlib и Pandas. Была изучена устойчивость работы данной нейронной сети при изменении следующих параметров: используемых оптимизаторов (Adam, Adamax и Rprop); коэффициента скорости обучения; количества нейронов в скрытом слое; количество эпох обучения. Делается вывод, что разработанная нейронная сеть может использоваться для моделирования уровня паводка при создании краткосрочных прогнозов. Для перехода в перспективе к более долговременным прогнозам предполагается в дальнейшем расширить размер факторов в обучающей выборке.

Ключевые слова

нейронные сети, прогнозирование, паводок, снежный покров, моделирование, PyTorch

Для цитирования

Шамсутдинова Т. М. Применение нейросетевого моделирования в задачах прогнозирования уровня паводка рек // Вестник НГУ. Серия: Информационные технологии. 2023. Т. 21, № 2. С. 39–50. DOI 10.25205/1818-7900-2023-21-2-39-50

Application of Neural Network Modeling in Problems of Predicting the Level of River Floods

Tatyana M. Shamsutdinova

Bashkir State Agrarian University
Ufa, Russia

tsham@rambler.ru, https://orcid.org/0000-0003-1809-3615

Abstract

The purpose of this article is to consider the theoretical and practical issues of developing neural network models for river flood forecasting (in case of the Belaya River near Ufa), as well as to implement the corresponding neural network

in Python. To build a training sample, archival data from meteorological services and meteorological observation sites for the flood periods of the Belaya (Agidel) River in 2018–2022 were used. The following indicators were collected and analyzed: water level, water temperature, day and night air temperature, precipitation, snow depth, including information about the pre-flood condition of the snow cover. The software implementation of the neural network was performed using the PyTorch deep learning library; in addition, modules from the Matplotlib and Pandas libraries were used. The stability of the operation of this neural network was studied when the following parameters were changed: the optimizers used (Adam, Adamax and Rprop); learning rate coefficient; the number of neurons in the hidden layer; number of learning epochs. It is concluded that the developed neural network can be used to model the flood level when creating short-term forecasts. In order to move to longer-term forecasts in the future, it is planned to further expand the size of the factors in the training sample.

Keywords

neural networks, forecasting, flood, snow cover, modeling, PyTorch

For citation

Shamsutdinova T. M. Application of Neural Network Modeling in Problems of Predicting the Level of River Floods. *Vestnik NSU. Series: Information Technologies*, 2023, vol. 21, no. 2, pp. 39–50. DOI 10.25205/1818-7900-2023-21-2-39-50

Введение

Ежегодные наводнения в результате весеннего разлива рек наносят колоссальный ущерб экономике страны, приводят к разрушению жилых домов, хозяйственных построек, дорожных покрытий, нарушают агротехнологический цикл сельскохозяйственного производства и т. д. В этих условиях большую актуальность приобретает задача прогнозирования уровня паводка рек, в том числе и с использованием методов нейросетевого моделирования.

Прогнозирование паводка рек является при этом очень сложной многопараметрической задачей, зависящей от большого числа разнообразных факторов природно-климатического характера, включая и техногенное воздействие человека. В частности, в качестве параметров нейросетевых моделей могут выступать следующие характеристики:

– количество выпадающих осадков, уровень почвенной влаги, уровень воды в реке, геометрические характеристики водосбора и водотока, параметры общей синоптической обстановки, атмосферного давления, ветра, а также параметры гидрологии всего русла [1];

– данные гидрометрических и дождемерных станций, вид рельефа, включая уклон и кривизну склона, накопление стока, тип почвы [2];

– данные систем спутникового мониторинга, например, размеры снежного покрова, температура поверхности и воздуха, водная маска поверхности воды и грунтовых вод, солнечное излучение, коэффициент испарения с поверхности [3];

– данные по подъему воды и градиенту гидротермического поля, включая показатель индикатора сезонности [4];

– влажность почвы на объекте в предпрогнозный период, замеры приращения уровня воды в паводковый период [5];

– количество осадков в верхней части водосбора и/или речного стока в верхних точках вдоль основных рек или притоков [6];

– паводкообразующие осадки и показатели предпаводочного увлажнения, зависящего от запаса воды в снежном покрове [7];

– рельеф местности, испарение влаги [8];

– параметры дождевого стока [9];

– предшествующие уровни воды реки в ее верховьях [10];

– сбросы воды на близлежащей ГЭС, состояние ледяного покрова на реке [11] и др.

Можно сказать, что в настоящее время задача применения нейронных сетей для прогнозирования уровня паводка носит научно-исследовательский характер. Существует ряд определенных проблем, связанных с получением требуемой эмпирической информации для обучения нейронных сетей, кроме этого, нет устоявшихся единообразных методик предобработки дан-

ных, выбора архитектуры нейронной сети – количества слоев, нейронов, функций активации, оптимизаторов и т. д. [12].

Цель данной статьи – рассмотреть теоретические и практические вопросы разработки нейросетевых моделей для прогнозирования паводка рек (на примере реки Белая в районе г. Уфы), а также реализовать нейронную сеть на языке Python с использованием библиотеки глубокого обучения PyTorch.

1. Материалы и методы

В ходе исследования были проанализированы данные наблюдений за паводковой ситуацией в бассейне реки Белая (Агидель) в районе г. Уфы.

При этом были использованы архивные данные метеослужб и сайтов метеонаблюдений за паводковые периоды реки Белая (Агидель) 2018–2022 годов:

– Архив уровней рек. Федеральное государственное бюджетное учреждение «Башкирское управление по гидрометеорологии и мониторингу окружающей среды». Пункт: г. Уфа. Водный объект: р. Белая. Выход на пойму: 660 см (www.meteorb.ru/arhiv-urovney-rek);

– Gismeteo. Дневник погоды в Уфе (www. www.gismeteo.ru/diary/4588/);

– World Weather. Погода в Уфе (www.world-weather.ru/pogoda/russia/ufa/);

– Расписание погоды. Архив погоды в Деме. Номер метеостанции 28722 (Башкортостан, Уфа) (www.rp5.ru/Архив_погоды_в_Деме).

Были собраны и проанализированы следующие показатели: уровень воды, температура воды, дневная температура воздуха, вечерняя температура воздуха, ночная температура воздуха, осадки, высота снежного покрова.

На рис. 1 представлены графики уровня весеннего паводка р. Белая в районе пункта г. Уфа в 2018–2022 годах. Как видим из графиков, динамика характера распределения локальных экстремумов паводка существенно различается по годам.

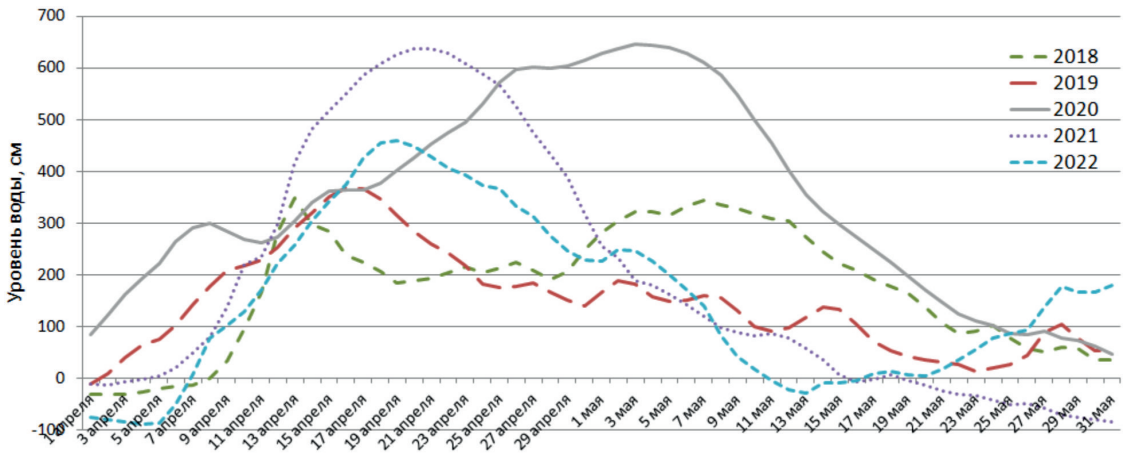


Рис. 1. Уровень паводка р. Белая в 2018–2022 годах
Fig. 1. Flood level of the Belaya River in 2018–2022

При этом данные имеют ряд выраженных экстремумов, что делает затруднительным процесс моделирования ряда с целью его дальнейшего прогнозирования. Ввиду этого было проведено частичное сглаживание рядов данных с помощью фильтра Ходрика–Прескотта в программном комплексе Loginom (с дальнейшим экспортом данных в электронные таблицы MS Excel). Описание некоторых особенностей применения фильтра Ходрика–Прескотта к задачам сглаживания нелинейных временных рядов приводится, например, в работе [13].

2. Результаты моделирования нейронной сети и ее обучения

Программная реализация нейронной сети выполнялась на языке Python с использованием библиотеки глубокого обучения PyTorch. Кроме этого, использовались модули библиотек Matplotlib и Pandas.

Начальная структура нейронной сети содержала три слоя, число нейронов в скрытом слое – 20, функция активации – Sigmoid, оптимизатор для выполнения шагов градиентного спуска – torch.optim.Adam (алгоритм Адама).

При этом был подготовлен массив данных (более 300 строк), используемый далее для обучения нейронной сети. Структура обучающего массива и его сводные показатели представлены в табл. 1 и 2.

Таблица 1

Сводные показатели по обучающей выборке
(средние значения по месяцам)

Table 1

Summary Indicators for the Training Sample (Monthly Averages)

Параметр (источник данных наблюдений)	2018 г.		2019 г.		2020 г.		2021 г.		2022 г.	
	апрель	май	апрель	май	апрель	май	апрель	май	апрель	май
Уровень воды, см на 8 часов (meteorb)	149	199	201	95	373	323	353	38	222	88
Температура воды, °С (meteorb)	1,6	8,6	3,8	10,5	3,8	11,1	2,4	12,1	2,6	9,5
Дневная температура воздуха, С (gismeteo)	7,9	16,0	10,7	18,3	9,3	18,9	11,9	24,6	11,6	14,6
Вечерняя температура воздуха, °С (gismeteo)	-0,1	7,0	0,4	8,9	3,0	7,9	10,2	23,5	10,1	13,6
Ночная температура воздуха, °С (world-weather)	0	7	0	9	3	8	3	11	4	6
Высота снежного покрова, см (rp5)	20,6	–	1,0	–	0,4	–	10,3	–	10,3	–

Таблица 2

Осадки, количество дней по месяцам
(по архивным данным сайта world-weather)

Table 2

Precipitation, Number of Days by Month
(According to the Archival Data from the World-Weather Website)

Вид осадков, (количество дней)	2018 г.		2019 г.		2020 г.		2021 г.		2022 г.	
	апрель	май	апрель	май	апрель	май	апрель	май	апрель	май
без осадков	23	26	27	28	22	28	28	29	23	22
слабый дождь	1	3	3	2	1	2	1	1	3	5
кратковременные осадки	5	2	0	1	6	1	0	1	4	4
сильный дождь	1	0	0	0	1	0	1	0	0	0

Ввиду большого влияния сформированного снежного покрова на уровень паводка рек было решено добавить в обучающую выборку также сведения о предпаводковом состоянии снежного покрова в первом квартале (январь – март) 2018–2022 годов (см. табл. 3).

Таблица 3

Высота снежного покрова, см
(по архивным данным https://rp5.ru, номер метеостанции 28722)

Table 3

Summary Snow Depth, cm
(According to the Archival Data https://rp5.ru, weather station number 28722)

Месяц	Сводные показатели по обучающей выборке	2018 г.	2019 г.	2020 г.	2021 г.	2022 г.
Январь	среднее	16,1	28,5	30,1	27,8	29,5
	максимальное	21	39	41	42	39
Февраль	среднее	19,1	47,4	40,9	31,3	42,7
	максимальное	36	54	48	42	51
Март	среднее	37,6	33,4	18,7	41,7	50,1
	максимальное	56	57	36	47	58

Как известно, после обучения на основе подготовленной выборки нейронная сеть приобретает возможность проводить расчеты и для данных, не используемых ранее в процессе обучения. Это позволяет использовать нейросетевые модели для задач прогнозирования данных.

На рис. 2 представлен полученный результат визуализации нейросетевой модели прогноза по данным об уровне реки Белая в районе г. Уфа за апрель–май 2022 г., включая продолжение прогнозного ряда на начало июня.

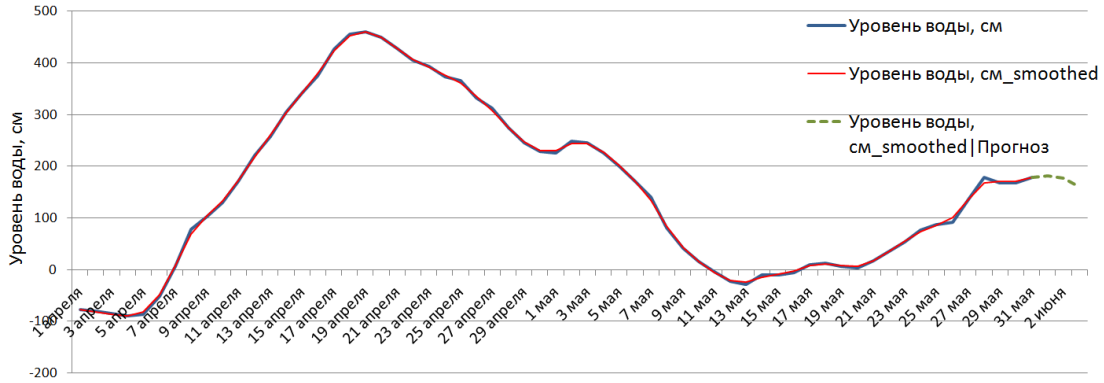


Рис. 2. Результаты нейросетевого моделирования (по данным за 2022 год)
Fig. 2. Results of neural network modeling (according to data for 2022)

Детализация данного прогноза представлена на рис. 3. Как видим на данном рисунке, в начале июня наблюдается спад уровня паводка с достаточно монотонно-гладкой динамикой уменьшения числовых показателей уровня воды.

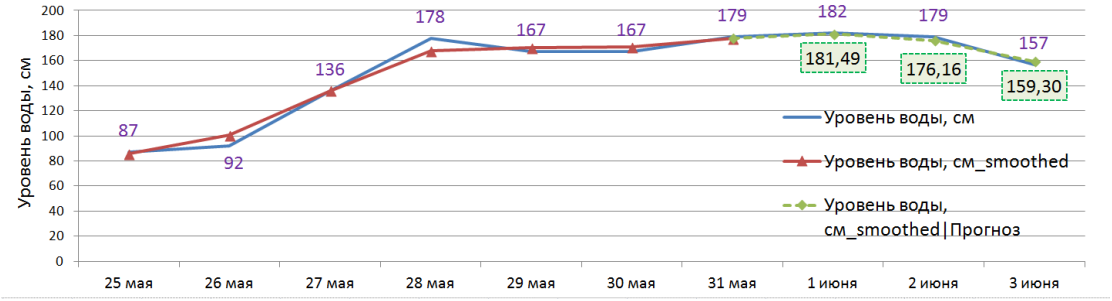


Рис. 3. Результаты прогноза на 1–3 июня 2022 года
Fig. 3. Forecast results for June 1–3, 2022

Для анализа качества полученных прогнозных значений были рассчитаны среднеквадратичная ошибка MSE (Mean Squared Error) по формуле (1) и средняя абсолютная ошибка в процентах MAPE (Mean Absolute Percentage Error) по формуле (2):

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - p_i)^2, \quad (1)$$

$$MAPE = \frac{100\%}{n}\sum_{i=1}^{n}\frac{|y_i - p_i|}{y_i}, \quad (2)$$

где y_i – эмпирическое значение, p_i – прогноз, n – количество измерений.

В результате расчетов получено значение MSE = 4.548, MAPE = 1,11 % что показывает хорошее качество прогноза в соотношении с размерностью исследуемых данных.

Но надо заметить, что прогнозирование такого рода функций, имеющих большое количество локальных экстремумов, представляет собой сложный процесс и вообще может быть успешно реализовано лишь для некоторых краткосрочных периодов. При увеличении периода прогнозирования (выше трех дней) наблюдалось увеличение показателей MSE и MAPE, из чего можно сделать вывод об эффективности применения данной сети именно для кратковременного прогнозирования.

3. Исследование параметров нейронной сети

Также была изучена устойчивость работы данной нейронной сети при изменении следующих параметров:

– используемых оптимизаторов;
– коэффициента скорости обучения lr;
– количества нейронов в скрытом слое;
– количество эпох обучения.

Был проведен сравнительный анализ динамики изменения функции потерь при выполнении шагов градиентного спуска для трех различных вариантов:

– torch.optim.Adam – алгоритм Адама;
– torch.optim.Adamax – алгоритм Адамакс (вариант Адама, основанный на норме бесконечности, *англ.* infinity norm);
– torch.optim.Rprop – устойчивый алгоритм обратного распространения.

Коэффициент скорости обучения при этом изменялся от lr = 0,001 до lr = 0,1; количество нейронов в скрытом слое варьировалось от 10 до 50.

Результаты экспериментального исследования по изучению динамики улучшения функции потерь (loss) представлены на рис. 4–6.

Как видим из рис. 4 и 5, при параметре скорости обучения lr = 0,01 увеличение числа нейронов скрытого слоя с 10 до 20 значительно увеличивает скорость метода градиентного спуска. Из рис. 6 видим, что увеличение параметра скорости обучения с lr = 0,01 до lr = 0,1 значительно уменьшает количество эпох сходимости оптимизаторов Adamax и Adam.

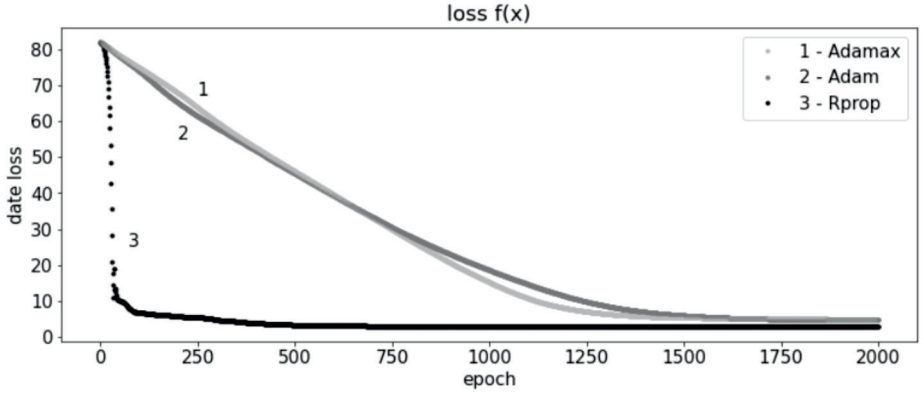


Рис. 4. График функции потерь в случае использования 20 нейронов в скрытом слое при скорости обучения lr = 0,01
Fig. 4. Graph of the loss function in the case of using 20 neurons in the hidden layer at a learning rate lr = 0.01

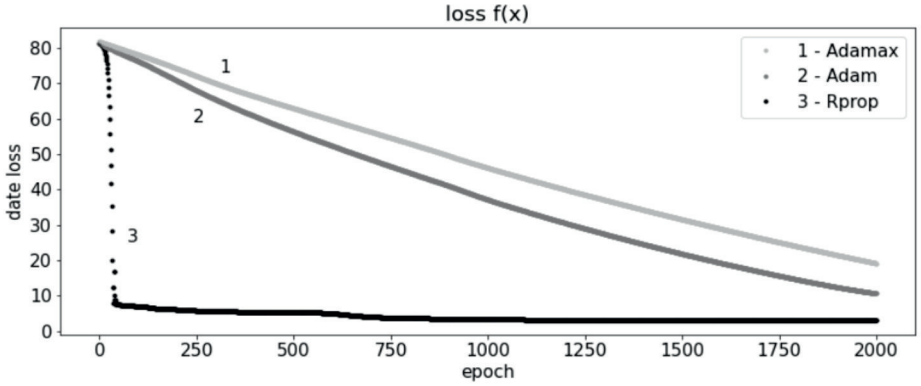


Рис. 5. График функции потерь в случае 10 нейронов в скрытом слое при скорости обучения lr = 0,01
Fig. 5. Graph of the loss function in the case of 10 neurons in the hidden layer at the learning rate lr = 0.01

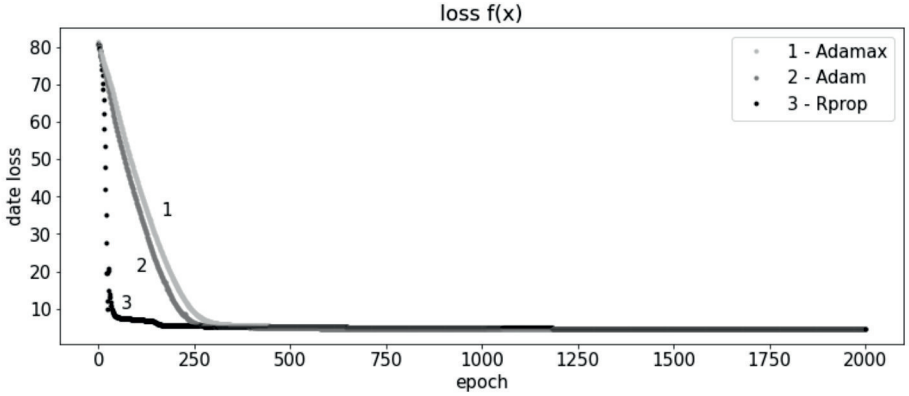


Рис. 6. График функции потерь в случае 10 нейронов в скрытом слое при скорости обучения lr = 0,1
Fig. 6. Graph of the loss function in the case of 10 neurons in the hidden layer at a learning rate lr = 0.1

Для параметра скорости обучения lr = 0,1 было получено, что для данного примера в качестве оптимального количества эпох обучения можно принять 1000 эпох, в ходе которых обеспечивается сходимость метода градиентного спуска для всех трех вариантов оптимизатора – Adam, Adamax и Rprop. При уменьшении скорости обучения до lr = 0,01 оптимизаторам Adam и Adamax в случае 10 нейронов в скрытом слое потребовалось более 3000 эпох обучения.

Также экспериментально было получено, что при уменьшении скорости обучения до lr = 0,001 очень существенно возрастает число эпох, необходимых для сходимости метода градиентного спуска. Уменьшения числа эпох можно при этом добиться с помощью увеличения числа нейронов в скрытом слое. Для данной модели при скорости обучения lr = 0,001 и количестве нейронов в скрытом слое равном 50 для сходимости градиентного спуска методами Adamax и Adam потребовалось около 5000 эпох.

Исходя из результатов численного исследования, можем сделать вывод об удовлетворяющем качестве построенной нейросетевой модели. Для различных вариантов количества нейронов в скрытом слое из диапазона от 10 до 50 удалось подобрать параметры обучения нейронной сети, обеспечивающие ее сходимость за менее чем 5000 эпох.

При этом можно сделать некоторые заключения, касающиеся сравнения оптимизаторов Adam, Adamax и Rprop.

В работе D. P. Kingma и J. Ba [14] отмечается, что метод Adam, относящийся к методам стохастической оптимизации, высоко эффективен в вычислительном отношении, не требователен к памяти, а также устойчив к диагональному изменению масштаба градиентов и хорошо подходит для задач с большим количеством параметров, например, для сверточных нейронных сетей (*англ*. Convolutional neural networks – CNNs). Там же говорится и о методе Adamax как одном из вариантов метода Adam, основанном на норме бесконечности. При этом уточняется, что в Adam правило обновления для отдельных весов заключается в масштабировании их градиентов обратно пропорционально масштабированной евклидовой норме L_2 их индивидуальных текущих и прошлых градиентов. При переходе к норме L_p в случае $p \to \infty$ получаем норму L_∞, определяемую как нахождение наибольшего из векторов в нормированном векторном пространстве. В результате применения данной нормы к расчету весовых коэффициентов нейронной сети получается новый устойчивый алгоритм оптимизации, получивший название Adamax.

Говоря о методе Rprop [15], C. Igel и M. Hüsken отмечают, что алгоритм Rprop (*англ*. resilient backpropagation – устойчивое обратное распространение) является одним из эффективных алгоритмов обучения первого порядка для нейронных сетей с произвольной топологией. Особенность обучения первого порядка характеризуется тем, что временная и пространственная сложность масштабируется только линейно в зависимости от количества оптимизируемых параметров. При этом C. Igel и M. Hüsken доказывают высокую эффективность метода для ряда примеров обучения нейронных сетей, в частности, для задач классификации данных и регрессионного анализа.

Что же можно сказать относительно сравнения оптимизаторов Adam, Adamax и Rprop применительно к данной конкретной построенной модели нейросетевого прогнозирования уровня паводка рек? В данном случае можем рассмотреть сравнение работы оптимизаторов с разных позиций: на основе количества эпох, требуемых для сходимости метода; на основе расчета среднеквадратичной ошибки MSE и средней абсолютной ошибки MAPE для полученного прогноза.

Как видим на рис. 4–6, во всех случаях метод Rprop показал более быструю сходимость градиентного спуска. Оптимизаторам же Adam и Adamax понадобилось большее, но сопоставимое между собой число эпох. Можно предположить, что быстрая сходимость метода градиентного спуска в случае применения оптимизатора Rprop обусловлена относительной несложностью топологии сети. Также отметим, что задача прогнозирования по своей концепции имеет некоторую близость к задачам регрессии, если принять во внимание, что алгоритмы

авторегрессионного анализа позволяют строить продолжение временного ряда с учетом его имеющихся предыдущих (ретроспективных) значений. Ввиду этого можно заключить, что полученные на рис. 4–6 результаты соответствуют изложенной в [15] концепции эффективности применения алгоритма Rprop для задач регрессионного анализа данных.

Методы Adam и Adamax ориентированы на объемные данные (в том числе на CNNs, используемые, например, при распознавании образов) и отличаются более точной подстройкой весовых коэффициентов за счет нелинейности применяемых в алгоритме моделей.

Как видим из табл. 4, построенной по исходным параметрам нейронной сети (число нейронов в скрытом слое – 20, функция активации – у оптимизатора Adamax.

Таблица 4

Сравнение прогнозов с фактическим уровнем паводка

Table 4

Comparison of Forecasts with Actual Flood Levels

Оптимизатор либо факт. уровень	Уровень на 01.06.2022, см	Уровень на 02.06.2022, см	Уровень на 03.06.2022, см	Среднеквадр. ошибка MSE	Средняя абс. ошибка MAPE
Adam	181,4900	176,1584	159,3045	4,54849788	1,11%
Adamax	182,0203	174,8007	158,2212	6,37528734	1,04%
Rprop	181,0734	174,3470	158,4074	8,16325711	1,33%
Уровень фактический	182	179	157	–	–

Но можем заметить, что все три варианта оптимизаторов Adam, Adamax и Rprop показали хорошее качество прогноза относительно средней абсолютной ошибки MAPE.

Выводы

Обобщая вышесказанное, можно сделать вывод, что существует ряд проблем, связанных с разработкой нейросетевых моделей для прогнозирования уровня паводка. Например, нехватка систематизированных данных гидротермических наблюдений, необходимых для обучения и тестирования модели. Кроме этого, на данный момент недостаточно изучен вопрос выбора оптимальной структуры нейросетевых моделей, включая выбор количества слоев, нейронов, видов функции активации, оптимизаторов и др. Отсутствует единая методика предобработки исходных эмпирических данных, связанная с их сглаживанием и устранением шумов и аномалий. Ввиду этого задача разработки и исследования новых нейросетевых моделей имеет большую востребованность и актуальность.

Разработанная нейронная сеть показала удовлетворяющие характеристики результатов моделирования при создании краткосрочных прогнозов, размер среднеквадратичной ошибки прогнозирования с использованием оптимизатора Adam составил MSE = 4,548 для прогноза на три дня. При увеличении периода прогнозирования (выше трех дней) наблюдалось увеличение показателя MSE, из чего можно сделать вывод об эффективности применения данной сети именно для кратковременного прогнозирования. Для перехода в перспективе к более долговременным прогнозам предполагается в дальнейшем расширить размер факторов в обучающей выборке, введя в нее дополнительные природно-климатические и ландшафтные характеристики, например, особенности рельефа местности, влияющие на удержание влаги в почве в районе русла рек и др.

Список литературы

1. **Афонин Л. А.** Проблемы прогнозирования паводков и наводнений // Наука. Инновации. Технологии. 2014, № 1. С. 145–152.
2. **Jahangir M. H., Reineh S. M. M., Abolghasemi M.** Spatial predication of flood zonation mapping in Kan River Basin, Iran, using artificial neural network algorithm. Weather and Climate Extremes, 2019, vol. 25, 100215. DOI: 10.1016/j.wace.2019.100215
3. **Ткаченко П. Н., Вакорин М .В.** Анализ проблемной ситуации использования информационных систем для прогнозирования паводка органами управления МЧС России // Сибирский пожарно-спасательный вестник. 2019, № 4 (15). С. 49–54.
4. **Варшанина Т. П., Митусов Д. В., Плисенко О. А., Стародуб И. В.** Нейросетевая модель прогноза паводков на малых реках Адыгеи // Известия Российской академии наук. Серия: Географическая. 2007, № 6. С. 87–93.
5. **Великанова Л. И.** Прогноз уровня воды при прохождении паводков на горных реках с применением нейросетевых технологий и прогноза метеослужбы // Проблемы автоматики и управления. 2013, № 1 (24). С. 66–73.
6. **Hag-Elsafi S.** Artificial Neural Networks (ANNs) for flood forecasting at Dongola Station in the River Nile, Sudan, Alexandria. Engineering Journal, 2014, vol. 53, issue 3, p. 655-662. DOI: 10.1016/j.aej.2014.06.010
7. **Лариошкин В. В.** Методика прогноза дождевых паводков в бассейне верхнего Амура (на примере р. Онон) // Известия Томского политехнического университета. Инжиниринг георесурсов. 2016. Т. 327, № 11. С. 105–115.
8. **Гребнев Я. В., Яровой А. В.** Мониторинг и прогнозирование паводков на территории Красноярского края использованием нейросетевых алгоритмов // Сибирский пожарно-спасательный вестник. 2018, № 3 (10). С. 13–16.
9. **Napolitano G., See L., Calvo B., Savi F., Heppenstall A.** A conceptual and neural network model for real-time flood forecasting of the Tiber River in Rome. Physics and Chemistry of the Earth, Parts A/B/C, 2010, vol. 35, issues 3–5, p. 187-194. DOI: 10.1016/j.pce.2009.12.004
10. **Castangia M., Grajales L.M.M., Aliberti A., Rossi C., Macii A., Macii E., Patti E.** Transformer neural networks for interpretable flood forecasting. Environmental Modelling & Software, 2023, vol. 160, 105581. DOI: 10.1016/j.envsoft.2022.105581
11. **Буянов Д. И., Федотов Р. С., Ткаченко П. Н.** Прогнозирование подъема уровня воды на реке Обь в Томской области на основе регрессионного анализа // Научные и образовательные проблемы гражданской защиты. 2015, № 2 (25). С. 112–118.
12. **Шамсутдинова Т. М.** Проблемы нейросетевого и регрессионного прогнозирования уровня паводка рек // Сибирский пожарно-спасательный вестник. 2021, № 2 (21). С. 99–105. DOI: 10.34987/vestnik.sibpsa.2021.79.13.017
13. **Li Z., Tanaka G.** HP-ESN: Echo State Networks Combined with Hodrick-Prescott Filter for Nonlinear Time-Series Prediction. 2020 International Joint Conference on Neural Networks (IJCNN), Glasgow, UK, 2020, p. 1-9. DOI: 10.1109/IJCNN48605.2020.9206771
14. **Kingma D. P., Ba J.** Adam: A Method for Stochastic Optimization. 3rd International Conference on Learning Representations (ICLR), San Diego, 2015. DOI: 10.48550/arXiv.1412.6980
15. **Igel C., Hüsken M.** Empirical Evaluation of the Improved Rprop Learning Algorithms. Neurocomputing, 2003, vol. 50, p. 105-123. DOI: 10.1016/S0925-2312(01)00700-7

References

1. **Afonin L. A.** Problems of forecasting floods and flooding. Science // Innovation. Technologies. 2014. No. 1. Pp. 145–152. (in Russ.)

2. **Jahangir M. H., Reineh S. M. M., Abolghasemi M.** Spatial predication of flood zonation mapping in Kan River Basin, Iran, using artificial neural network algorithm // Weather and Climate Extremes. 2019. Vol. 25. 100215. DOI 10.1016/j.wace.2019.100215
3. **Tkachenko P. N., Vakorin M. V.** Analysis of the problem situation in case of using information flood forecasting systems by units of EMERCOM of Russia // Siberian Fire and Rescue Bulletin. 2019. No. 4(15). Pp. 49–54. (in Russ.)
4. **Varshanina T. P., Matusov D. V., Plisenko O. A., Starodub I. V.** Neuro-Network System of the Flood Forecast on Adygei Small Rivers // Izvestiya Rossiiskoi Akademii Nauk. Seriya Geograficheskaya. 2007. No. 6. Pp. 87–93. (in Russ.)
5. **Velikanova L. I.** Forecasting the Water Level at Forwarding Flood in The Mountain Streams Using Neural Network Technology and Forecast of Meteorological Service. Automation and Control Problems. 2013. No. 1(24). Pp. 66–73. (in Russ.)
6. **Hag-Elsafi S.** Artificial Neural Networks (ANNs) for flood forecasting at Dongola Station in the River Nile, Sudan, Alexandria // Engineering Journal. 2014. Vol. 53, iss. 3. Pp. 655–662. DOI 10.1016/j.aej.2014.06.010
7. **Larioshkin V. V.** Technique of Forecasting Rain Flood in the Upper Amur Basin (by the Example of the Onon River) // Bulletin of the Tomsk Polytechnic University. Geo Assets Engineering. 2016. Vol. 327. No. 11. Pp. 105–115. (in Russ.)
8. **Grebnev Y. V., Yarovoy A. V.** Control and prediction of floods on the territory of the Krasnoyarsk kray through the use of neural network algorithms // Siberian Fire and Rescue Bulletin. 2018. No. 3(10). Pp. 13–16. (in Russ.)
9. **Napolitano G., See L., Calvo B., Savi F., Heppenstall A.** A conceptual and neural network model for real-time flood forecasting of the Tiber River in Rome // Physics and Chemistry of the Earth, Parts A/B/C. 2010. Vol. 35. Iss. 3–5. Pp. 187–194. DOI 10.1016/j.pce.2009.12.004
10. **Castangia M., Grajales L. M. M., Aliberti A., Rossi C., Macii A., Macii E., Patti E.** Transformer neural networks for interpretable flood forecasting // Environmental Modelling & Software. 2023. Vol. 160. 105581. DOI 10.1016/j.envsoft.2022.105581
11. **Buyanov D. I., Fedotov R. S., Tkachenko P. N.** Prediction of the Ob's Water Level Rise in Tomsk Region Using Regression Analysis // Scientific and educational problems of civil protection. 2015. No. 2(25). Pp. 112–118. (in Russ.)
12. **Shamsutdinova T. M.** Problems of neural network and regression forecasting the river flood level // Siberian Fire and Rescue Bulletin. 2021. No. 2(21). Pp. 99–105. (in Russ.) DOI 10.34987/vestnik.sibpsa.2021.79.13.017
13. **Li Z., Tanaka G.** HP-ESN: Echo State Networks Combined with Hodrick-Prescott Filter for Nonlinear Time-Series Prediction // 2020 International Joint Conference on Neural Networks (IJCNN), Glasgow, UK, 2020. Pp. 1–9. DOI doi.org/10.1109/IJCNN48605.2020.9206771
14. **Kingma D. P., Ba J.** Adam: A Method for Stochastic Optimization // 3rd International Conference on Learning Representations (ICLR), San Diego, 2015. DOI 10.48550/arXiv.1412.6980
15. **Igel C., Hüsken M.** Empirical Evaluation of the Improved Rprop Learning Algorithms // Neurocomputing. 2003. Vol. 50. Pp. 105–123. DOI 10.1016/S0925-2312(01)00700-7

Информация об авторе

Шамсутдинова Татьяна Михайловна, кандидат физико-математических наук, доцент кафедры цифровых технологий и прикладной информатики Башкирского государственного аграрного университета

Information about the Author

Tatyana M. Shamsutdinova, Candidate of Sciences (Physics and Mathematics), Docent of the Department of Digital Technologies and Applied Informatics, Bashkir State Agrarian University

Статья поступила в редакцию 04.04.2023;
одобрена после рецензирования 07.07.2023; принята к публикации 07.07.2023

The article was submitted 04.04.2023;
approved after reviewing 07.07.2023; accepted for publication 07.07.2023

Научная статья

УДК 005.13.11
DOI 10.25205/1818-7900-2023-21-2-51-62

Проблемы оценки качества архитектур нейронных сетей и алгоритмов поиска архитектур

Андрей Сергеевич Щербин

Новосибирский государственный научный исследовательский университет
Новосибирск, Россия

a.shsherbin@g.nsu.ru

Аннотация

В статье приведен обзор научных публикаций в области поиска архитектур нейронных сетей. Рассмотрены не только публикации, посвященные алгоритмам поиска архитектур, но и серия статьей, посвященная оценке качества алгоритмов поиска. На основе проведенного обзора обозначены актуальные проблемы в области оценки качества архитектур нейронных сетей и сравнения алгоритмов поиска.

Ключевые слова

нейронные сети, поиск архитектур нейронных сетей, оценка качества алгоритмов поиска архитектур

Для цитирования

Щербин А. С. Проблемы оценки качества архитектур нейронных сетей и алгоритмов поиска архитектур // Вестник НГУ. Серия: Информационные технологии. 2023. Т. 21, № 2. С. 51–62. DOI 10.25205/1818-7900-2023-21-2-51-62

Problems of Neural Network Architecture Benchmarking and Search

Andrey S. Shcherbin

Novosibirsk State University
Novosibirsk, Russian Federation

a.shsherbin@g.nsu.ru

Abstract

In this paper we made a survey of neural architecture search algorithms and their benchmarking. Based on our survey we highlight the current problems in the quality of neural network architecture benchmarking and in the comparison of neural architecture search algorithms.

Keywords

neural networks, neural architecture search, neural architecture search algorithms benchmarking

For citation

Shcherbin A. S. Problems of Neural Network Architecture Benchmarking and Search. *Vestnik NSU. Series: Information Technologies*, 2023, vol. 21, no. 2, pp. 51–62. DOI 10.25205/1818-7900-2023-21-2-51-62

Введение

Темпы развития алгоритмов искусственного интеллекта, основанного на искусственных нейронных сетях, показывают, что их можно считать достаточно универсальной моделью для решения разного рода практических задач. Однако вместе с универсальностью эти модели имеют и ряд недостатков, в их числе вычислительная сложность, а также объем требуемой для вычислений памяти. Эти особенности ведут к увеличению времени отклика приложений, в основе которых используются нейронные сети, а также к увеличению энергопотребления систем на их основе.

Скорость работы и потребление памяти в приложениях, основанных на нейронных сетях, чаще всего определяется архитектурой сети. Задачу нахождения оптимальной архитектуры сети с точки зрения требуемых вычислительных ресурсов, а также метрики качества решают методами поиска архитектур нейронных сетей (Neural architecture search). Для поиска применяются различные алгоритмы оптимизации: эволюционные [1], байесовские [2], дифференцируемые [3], а также алгоритмы, основанные на обучении с подкреплением [4, 5].

Задача поиска архитектуры сети является задачей оптимизации с ограничениями. Ограничения зачастую накладываются специалистами-практиками, и могут зависеть от объема памяти в целевом устройстве, требований бизнеса на время отклика системы, количества арифметико-логических устройств в разрабатываемом процессоре. То есть задача состоит в том, чтобы найти архитектуру с заданной вычислительной сложностью (возможно не ограниченной), имеющую максимальную точность для конкретной задачи.

Архитектура нейронной сети может быть представлена в виде графа, вершинами которого являются операции, а ребрами – зависимости по данным. Поиск архитектуры требуется завершить за конечное время, для этого исследователи ограничивают набор возможных операций, называя полученное множество возможных архитектур пространством поиска.

Оценка качества алгоритмов поиска архитектур является вычислительно сложной задачей, так как требует обучения всех возможных архитектур из пространства поиска. Более того, так как обучение нейронной сети является стохастическим процессом, а его результат зависит от множества гиперпараметров, проблема оценки качества алгоритмов поиска дополнительно усложняется.

1. Постановка задачи поиска архитектуры нейронных сетей

Пусть Θ – множество всех возможных архитектур, заданное исследователем; λ – функция оценки вычислительной сложности архитектуры нейронной сети; γ – верхняя граница вычислительной сложности архитектуры; μ – функция оценки качества работы архитектуры на решаемой задаче. Тогда задача поиска архитектуры может быть сформулирована следующим образом:

$$\alpha = \max(\mu(\alpha i)) \mid \lambda(\alpha i) <= \gamma \forall \alpha i \in \Theta.$$

Сложность получения $\lambda(\alpha)$ на практике зависит от того, на какие параметры накладываются ограничения. Часто необходимо, чтобы модель работала с заданной скоростью на заданном устройстве. В таком случае для оценки времени работы производится измерение этого времени на целевом устройстве с учетом всех оптимизаций программной платформы, а также особенностей устройства. Программная платформа может выбирать различные алгоритмы вычисления операций в зависимости от архитектуры модели: занимаемой ею памяти, степени параллелизма операций, количества свободных арифметико-логических устройств в каждый момент вычисления нейронной сети. В устройстве могут быть эффективно реализованы опе-

рации для конкретных размерностей тензоров входных данных или для конкретных видов операций (например свертки, количество каналов которых кратно 8).

Во многих методах [6, 7, 8] в оптимизируемый функционал добавляются дополнительные ограничения, позволяющие находить архитектуры, удовлетворяющие требованиям не только по качеству работы модели, но и по потребляемой памяти, и времени ее вычисления. Архитектуры [8, 9], найденные для задачи ImageNet2012 [10], показывают лучшее качество при меньшем количестве операций, т. е. являются более эффективными, чем разработанные исследователями вручную с точки зрения энергопотребления. Исследователи в области поиска архитектур часто анализируют результаты своих работ в терминах Парето оптимальности. Архитектура считается оптимальной по Парето в некотором пространстве критериев, если ее улучшение по какому-либо критерию ведет к ухудшению по другим критериям.

Стоит отметить, что реализация новых идей в архитектурах нейронных сетей (остаточные связи, групповые свертки, отдельная обработка пространственных и поканальных признаков) итеративно увеличивала качество моделей на задаче ImageNet. Однако могут существовать более оптимальные (по различным критериям) модели для этой задачи, реализующие те же самые идеи. Оптимальность архитектуры достигается за счет подбора гиперпараметров под конкретную задачу.

2. Обзор алгоритмов поиска архитектур

Для понимания развития направления поиска архитектур нейронных сетей необходимо рассмотреть несколько знаковых работ. В большинстве из них основной задачей является поиск архитектуры, оптимальной для конкретной задачи, однако не производится детального анализа свойств предложенного алгоритма поиска. Некоторые из предложенных алгоритмов являются вычислительно сложными, что препятствует их применению в практических задачах.

2.1. Поиск архитектур через обучение с подкреплением

В первых работах [4], посвященных поиску архитектур нейронных сетей, поиск производился при помощи нейронной сети-контроллера, состоящей из ячеек с долгой краткосрочной памятью (Long Short Term Memory cells) [11] и алгоритма обучения с подкреплением REINFORCE [12]. Исследователи решали задачу нахождения архитектуры с максимальной точностью классификации на наборе данных CIFAR-10 [13]. Для каждой архитектуры-кандидата производилась тренировка модели с целью оценить ее точность. Для вычислительного эксперимента использовались 800 графических ускорителей. Для тренировки всех архитектур-кандидатов использовался одинаковый набор гиперпараметров обучения, предложенный в статье [14]. Гиперпараметры обучения сети-контроллера были подобраны исследователями самостоятельно.

Авторы предложили подход к поиску архитектур нейронных сетей, который получил развитие, однако при использовании одинаковых гиперпараметров обучения для оценки качества всех архитектур постановка задачи сужается. Различные архитектуры могут иметь различные оптимальные гиперпараметры обучения для одной и той же задачи, что вносит некоторый сдвиг в функцию оценки качества архитектуры при применении одинаковых гиперпараметров. Вследствие наличия сдвига утверждать об оптимальности архитектуры для задачи возможно только в контексте конкретных гиперпараметров обучения.

2.2. Алгоритм NASNet

Развитием описанных выше идей является алгоритм, который позволил найти одну из популярных по сей день сгенерированных архитектур, – NASNet [5]. В этой статье исследователи

заранее определили структуру нейронной сети: последовательность и количество блоков двух видов. Обычный блок (Normal block) не изменял пространственного разрешения карты признаков, сжимающий блок (Reduction block) уменьшал пространственное разрешение. Задача поиска архитектуры была сформулирована как поиск оптимальной структуры каждого из блоков. Поиск производился в ограниченном пространстве операций: свертки с разными размерами ядер, операции подвыборки (pooling) с разными размерами ядер, поканальные и разреженные свертки, а также тождественная операция. В качестве алгоритма поиска использовалась рекуррентная нейронная сеть, называемая контроллером. Архитектура представлялась в виде последовательности символов, что позволило авторам использовать для поиска архитектуры тот же подход, что и для генерации текста. Для оценки качества предложенного подхода авторы обращают внимание читателя на точность найденной нейронной сети на наборе данных ImageNet и утверждают, что найденная архитектура обладает лучшей точностью, чем текущие лучшие решения, при меньших вычислительных затратах.

Однако подход, которым была найдена архитектура NASNet, является вычислительно сложным, так как производит обучение сети-контроллера. Также данный подход использует предположение о том, что существуют блоки, которые являются оптимальными на любом этапе обработки данных. Данное предположение уменьшает пространство поиска архитектур. Такой подход называется микропоиском, он противопоставляется макропоиску, когда каждый блок в сети может быть составлен из разных операций и количество используемых блоков может быть найдено алгоритмом. Макропоиск является еще более вычислительно сложным, так как количество возможных архитектур экспоненциально возрастает в сравнении с микропоиском, и для больших структур некоторые алгоритмы могут быть подвержены проклятию размерности.

2.3. PNASNet

Метод итеративного (от простого к сложному) макропоиска архитектур предложен в [15]. Идея заключается в усложнении пространства поиска в процессе оптимизации путем добавления дополнительных ячеек. Структура ячейки является фиксированной, однако набор операций в ней оптимизируется. При этом каждая ячейка может иметь собственный набор операций, оптимальный для конкретной глубины сети. В процессе оптимизации исследователи обучают модель для оценки качества архитектур. Это позволяет сократить требуемые вычислительные ресурсы, а также производить выбор операций для ячеек с учетом информации о влиянии предыдущих операций-кандидатов на точность модели.

Модель для оценки качества по архитектуре обучается на первых двух ячейках, при добавлении последующих ячеек их архитектуры ранжируются в соответствии с предсказаниями этой модели. Нейронные сети с наиболее перспективными по предсказаниям модели-оценщика ячейками дообучаются на целевой задаче, что позволяет дообучить и модель-оценщика.

В качестве модели-оценщика авторы использовали нейронную сеть с долгой краткосрочной памятью [11], архитектура и гиперпараметры модели отличались от предложенных в [5], что не позволяет сделать вывод о вкладе метода итеративного поиска в итоговую точность найденной модели и сравнить алгоритмы поиска в одинаковых условиях.

3. Развитие методов оценки качества алгоритмов поиска

Проблема воспроизводимости результатов исследований, посвященных поиску архитектур нейронных сетей, поднимается во многих публикациях [16, 17, 18, 19, 20, 21]. Исследователи отмечают, что причинами проблем с воспроизводимостью являются различия в постановке задачи, выборе пространства поиска, способах оценки качества моделей-кандидатов, гиперпараметрах обучения. Для корректного сравнения алгоритмов поиска архитектур независимо

от гиперпараметров обучения, пространства поиска и постановки задачи исследователи проводят обучение всех возможных архитектур с различными гиперпараметрами из некоторого множества конфигураций. Далее предлагается использовать в алгоритме поиска точность обученной архитектуры в качестве оценки ее качества. Далее рассмотрим реализации этой идеи, их преимущества и недостатки.

3.1. NASBench-101

Рассмотрим первой статью NasBench-101 [17] для того, чтобы более детально сформировать представление о предлагаемом подходе, а также понять мотивацию принимаемых авторами решений. Основная идея статьи состоит в создании набора данных, содержащего точности решения задачи классификации для каждой из архитектур из пространства поиска. Этот набор данных предлагается использовать как разметку, а также для оценки качества предлагаемых алгоритмом в процессе поиска архитектур. Обучение моделей производилось с одним набором гиперпараметров для всех архитектур на наборе данных CIFAR-10.

Задача поиска заключалась в оптимизации структуры блока нейронной сети, который использовался в заранее определенной структуре. То есть задача представляла собой микропоиск, что было обосновано вычислительной сложностью. Архитектура блока представлялась исследователями в виде графа, вершинами которого были операции, а ребрами – зависимости по данным. Также для ограничения пространства поиска были введены некоторые ограничения: допускалось использование только трех операций, число вершин было ограничено семью, а число ребер в графе 9.

Исследователи продемонстрировали Парето-фронт при помощи графиков, определили понятие локальности архитектур как расстояние редактирования представления графа. В результате оценки автокорреляции архитектур методом случайных блужданий было показано, что корреляция становится отличима от шума при расстоянии большем или равном шести. Также авторы заметили, что 35 % всех архитектур находятся на расстоянии редактирования не более чем 6 от лучшей выбранной архитектуры. Это означает, что 1 из 50 000 случайно выбранных архитектур является неотличимой от наилучшей архитектуры.

Для оценки качества авторы выбрали алгоритмы поиска архитектур, основанные на байесовских методах, эволюционных алгоритмах, обучении с подкреплением, оценке качества моделей в процессе их обучения, а также случайном выборе. Методы сравнивались в условиях ограниченного времени на поиск оптимальной архитектуры. Так как время на выполнение одной итерации отличается для разных методов, было произведено различное количество итераций. В качестве результатов сравнения алгоритмов авторы приводят следующие выводы:

- Эволюционный алгоритм, а также некоторые байесовские методы на данной задаче сходятся в 5 раз быстрее, чем метод случайного выбора архитектур.
- Алгоритм, использующий обучение с подкреплением хоть в начале и превосходил по качеству алгоритм случайного выбора, но в итоге уступил ему, так как требовал больших вычислительных ресурсов на итерацию.

3.2. NAS-Bench201

Следующая статья [18], посвященная воспроизводимости результатов и сравнимости методов в области поиска архитектур нейронных сетей продолжает идеи NasBench-101. Одним из методологических отличий является отказ от ограничений на число связей в вычислительном графе. Это позволяет расширить множество применимых алгоритмов поиска архитектур градиентными методами.

Также авторы данной статьи изменили представление вычислительного графа: в качестве вершин теперь выступают данные, а операции кодируются ребрами графа. Структура блока,

а также набор возможных операций в пространстве поиска были выбраны таким образом, чтобы в пространство поиска входила архитектура ResNet. Таким образом, исследователям удалось уменьшить количество уникальных архитектур блока в пространстве поиска за счет сокращения количества возможных операций и связей между ними.

Для более подробного изучения алгоритмов поиска в качестве одного из результатов исследователи представили историю тренировки для каждой из архитектур пространства поиска. Это позволяет оценивать качество алгоритмов, в которых оценка качества архитектуры-кандидата происходит путем ее тренировки некоторое количество шагов. Обучение моделей в рамках данной работы производилось на трех наборах данных: CIFAR-100, CIFAR-10 и ImageNet-16-120. Последний набор данных был получен из ImageNet путем выбора первых 120 классов и уменьшения разрешения изображений до 16 х 16. Возможность оценить качество модели на нескольких задачах также позволяет получить более устойчивую оценку качества алгоритма поиска. К тому же наличие нескольких задач позволяет производить поиск оптимальной архитектуры на одном наборе данных, а сравнение и оценку точности проводить на других.

Исследователи по-прежнему использовали одинаковые гиперпараметры для обучения всех архитектур. Гиперпараметры были выбраны на основе анализа научных статей, использующих наборы данных CIFAR. Для ImageNet-16-120 использовался тот же набор гиперпараметров, что и для CIFAR-100. Такой подход позволяет сократить количество обучений моделей и моделирует использование исследователем фиксированного набора гиперпараметров при изменении архитектуры модели, что имеет место в практических приложениях. Результатом исследования является набор данных, который содержит информацию о ходе каждого из трех обучений каждой модели из описанного пространства поиска на каждую из трех задач классификации.

В заключение авторы приводят ряд рекомендаций по использованию набора данных NasBench-201:

– Избегать переобучения под NasBench-201:
  - Не использовать специализированных ограничений на операции в пространстве поиска. Например, не ограничивать количество операций определенного типа.
  - Использовать при оценке кандидатов и выборе наилучшей модели представленные в наборе данных значения качества, даже если имеется другая конфигурация гиперпараметров, позволяющая получить лучшее качество.
  - В качестве итоговой метрики качества алгоритма поиска использовать результат нескольких запусков. Это делает результаты более устойчивыми, и не требует больших вычислительных затрат при использовании набора данных NasBench.
– Помнить о возможном шуме в оценке качества архитектур, вызванном использованием одинаковых конфигураций гиперпараметров для разных архитектур. Возможным решением может быть оптимизация гиперпараметров, однако это требует дополнительных вычислительных ресурсов.
– Не использовать алгоритмы поиска архитектур, устройство которых может вносить шум в оценку качества моделей. Например, алгоритмы с разделяемыми весами, в которых поиск заключается в выборе оптимальной подсети. Методики обучения таких алгоритмов могут влиять на точность всех моделей в пространстве поиска reinforce.
– Оценка генерализации алгоритмов поиска архитектур на наборе данных NasBench-201 выглядит следующим образом: эволюционный алгоритм > алгоритм reinforce > случайный выбор. Данный порядок совпадает с результатами, полученными на наборе данных NasBench-101. Порядок в оценке генерализации градиентных методов поиска: GDAS [22] > DARTS [3] > ENAS [23] совпадает с оценками, полученными в статье NasBench1shot1 [19]. Значит, результаты, полученные на NasBench-201, могут генерализироваться на другие наборы данных для оценки качества алгоритмов поиска.

3.3. HW-NAS-Bench

С целью расширения границ применимости метода оценки, предложенного авторами [18], авторы [20] расширили имеющуюся базу данных. Основная задача работы заключалась в предоставлении возможности исследователям в области поиска архитектур нейронных сетей, зачастую не имеющим компетенций в работе с устройствами на низком уровне, оценить качество разрабатываемых алгоритмов в условиях ограниченного времени исполнения найденной модели на конечном устройстве.

Для решения поставленной задачи авторы измерили время вычисления нейронных сетей из пространства поиска [18] на устройствах разных классов:

- Компактный графический ускоритель (Nvidia Jetson TX2);
- Одноплатный компьютер для интернета вещей (Raspberry Pi 4);
- Тензорный процессор, разработанный специально для вычисления нейронных сетей (Google TPU);
- Мобильный телефон, имеющий специализированный сопроцессор (Google Pixel 4);
- Интегральная схема специального назначения (ASIC-Eyeriss).

По результатам измерений времени вычисления различных операций исследователи обнаружили, что для разных программно-аппаратных платформ оптимальными являются разные архитектуры. На основе полученного авторами результата можно сделать вывод, что косвенные метрики сложности вычисления модели, например, количество операций сложения и умножения, являются не репрезентативными, если задача состоит в ускорении модели на конкретном устройстве.

Также особенностью методологии, описанной в статье, является использование, помимо пространства поиска для блока нейронной сети, и пространства поиска, которое позволяет находить различные блоки на разных этапах обработки данных. Авторы использовали пространство поиска, описанное в статье [7], которое в оригинальной реализации содержит порядка 10^{23} архитектур. Для этого пространства поиска авторы измерили время вычисления операций, и воспользовались предположением, что сумма времен вычисления операций равна времени вычисления модели. Данное предположение было проверено на 100 случайных архитектурах из пространства [7], и подтверждено высокими коэффициентами корреляции.

Таким образом, авторы привнесли в методологию разработки метода оценки качества алгоритмов поиска привязку эффективности найденных архитектур нейронных сетей к конкретным программно-аппаратным комплексам и создали возможность производить такую оценку эффективности моделей для устройств разных классов. Это может стать фундаментом для появления и развития алгоритмов поиска архитектур нейронных сетей, специализированных под конкретные программно-аппаратные решения.

Однако результаты, описанные в статье, трудно воспроизводимы, так как авторы не зафиксировали версии программного обеспечения, которое было установлено на устройствах во время выполнения измерений.

3.4. NAS-HPO-BENCH

В работе [21] исследователи обращают внимание на гиперпараметры обучения моделей. Основная задача авторов – создание базы данных, в которой сохранена информация об обучении моделей разных архитектур с различными гиперпараметрами. Такая база данных может использоваться другими исследователями для оценки качества собственных алгоритмов поиска гиперпараметров и архитектур моделей. Также на основе данных, собранных в базе, возможен анализ важности гиперпараметров, относящихся к обучению и архитектуре моделей.

Обучение моделей производилось на четырех табличных наборах данных, различных по количеству примеров и количеству признаков в таблице. Однако для всех наборов дан-

ных решалась задача регрессии. В качестве архитектурных гиперпараметров выступали: количество нейронов, вероятность случайного отключения нейронов (dropout), а также функция активации для каждого из двух слоев нейронной сети. В качестве параметров обучения были выбраны: начальный шаг обучения (learning rate), размер пакета для оптимизации (batch size), расписание шага обучения (learning rate scheduler). В общей сложности было обучено 995328 модели, по 62208 моделей для каждого из четырех наборов данных. При этом каждая конфигурация обучалась четыре раза для возможности оценки шума. Каждая модель обучалась в течение 100 эпох при помощи алгоритма оптимизации Adam [24].

В результате анализа собранной в базе данных информации исследователи выяснили, что шум в значениях средней квадратичной ошибки уменьшается с увеличением количества эпох обучения. Для большинства моделей основным параметром по оценке важности методом Functional ANOVA оказался начальный шаг обучения. Однако оценка всех конфигураций не является информативной, так как в практических экспериментах специалистов чаще всего интересуют модели с наименьшей ошибкой. Поэтому был проведен анализ важности гиперпараметров для 1 % и 10 % моделей с наименьшей ошибкой. Также с целью учесть возможные взаимосвязи между гиперпараметрами была проведена оценка важности пар гиперпараметров. По результатам оценки оказалось, что для 1 % моделей с наименьшей ошибкой наиболее важными параметрами является начальный шаг обучения и количество нейронов в первом слое нейронной сети. А для 10 % наиболее точных моделей – начальный шаг обучения и функция активации первого слоя.

Следующим шагом исследователи провели сравнение существующих методов оптимизации гиперпараметров на полученной базе данных. Методы, основанные на байесовской оптимизации, а именно SMAC, TPE и Bohamiann на первых итерациях показывали качество, сравнимое со случайным выбором значений. Однако после накопления статистики, которое произошло примерно в одно время, стали превосходить его. Перечисленные методы при этом достигли разных оптимумов, что объясняется различными моделями (априорными распределениями), используемыми в методах.

Алгоритмы с возможностью оптимальным образом использовать вычислительные ресурсы: Hyperband (HB), Bayesian optimization hyperband (BOHB) показали такое же поведение, как и байесовские методы. В BOHB для оценки полезности примеров использовалась функция, учитывающая взаимосвязи между конфигурациями гиперпараметров, в отличие от TPE. Однако TPE превзошел BOHB и HB по качеству, что исследователи связывают с использованием в BOHB параметров по умолчанию, которые могут быть не оптимальны в данной конкретной задаче, однако превосходят TPE на большом наборе задач [25].

Эволюционному алгоритму потребовалось еще больше итераций, чтобы превзойти качество случайного поиска. Но именно он достиг наилучших результатов из всех оцениваемых алгоритмов. Алгоритм, основанный на обучении с подкреплением, оказался слишком неэффективным по количеству семплирований и лишь немного лучше случайного поиска.

3.5. NAS-HPO-BENCH II

В следующей статье, посвященной оценке алгоритмов поиска архитектур и гиперпараметров обучения моделей [26], авторы используют набор данных CIFAR-10 и поиск оптимальной архитектуры сверточной нейронной сети. В качестве гиперпараметров обучения в данной работе исследователи используют размер пакета (batch size) для оптимизации и шаг обучения (learning rate). При этом производится поиск оптимального блока сети в пространстве из 4 000 вариантов. Авторы выполнили обучение каждой конфигурации 3 раза в течение 12 эпох и обучили модель предсказывать качество после 200 эпох обучения. Итого набор данных об обучениях моделей, который предлагается использовать для оценки качества алгорит-

мов поиска, содержит 192 000 комбинаций гиперпараметров и архитектур и данные о каждом из трех таких обучений.

Для предсказания качества моделей после 200 эпох тренировки авторы обучили еще 4 800 конфигураций. Эти конфигурации использовались как набор данных для обучения суррогатной модели. Структура блока нейронной сети превращалась в вектор при помощи сети изоморфизма графа [27], а представление выбранных параметров обучения как бинарных векторов – при помощи многослойного перцептрона. Полученные векторы параметров обучения и архитектуры модели конкатинировались и подавались на вход другому многослойному персептрону.

Суррогатная модель являлась объединением 10 многослойных перцептронов, обученных на разных подмножествах конфигураций. Также производилась кросс-оценка качества модели на 5 непересекающихся подмножествах. В результате была получена модель, которая решает задачу предсказания качества по архитектуре, параметрам обучения и качеству на двенадцатой эпохе через 200 эпох обучения с коэффициентом детерминации 0,876. Исследователи рекомендуют использовать для предсказания точности после 200 эпох ансамбль моделей, обученных на пяти используемых подмножествах.

В качестве алгоритмов поиска гиперпараметров для тестирования использовалось 6 алгоритмов. Метрикой для оценки качества являлась ошибка на тестовом множестве найденной модели (выбранной архитектуры, обученной с выбранными гиперпараметрами). Приведем алгоритмы в порядке улучшения качества:

1. Байесовский поиск параметров обучения с остановкой обучения наименее перспективных конфигураций и поиск архитектуры через регуляризованную эволюцию;
2. Случайный поиск параметров обучения и поиск архитектуры через регуляризованную эволюцию;
3. Случайных поиск;
4. Алгоритм, основанный на обучении с подкреплением reinforce [12];
5. Байесовский поиск архитектуры и параметров обучения с остановкой обучения наименее перспективных конфигураций;
6. Регуляризованная эволюция.

С этой работы начинаются значительные изменения в методологии оценки качества алгоритмов поиска архитектур (и подбора гиперпараметров), так как здесь появляется идея использования модели для предсказания точности конфигурации. Использование этой идеи позволяет сократить вычислительную сложность разработки наборов данных с информацией о качестве моделей, но в то же время может приводить к неточностям в оценках качества самих алгоритмов поиска.

4. Проблемы оценки качества алгоритмов поиска архитектур нейронных сетей

Современные методы поиска архитектур [6, 7, 28] предполагают извлечение оптимальной подсети из большой обученной модели. Это позволяет снизить стоимость поиска, так как позволяет использовать одну обученную модель для извлечения подсетей с разными характеристиками. При этом происходит глобальный поиск архитектуры сети, т. е. в разных частях модели могут выбраться блоки разной архитектуры. Для оценки качества таких алгоритмов требуются свои наборы данных, отличные от рассмотренных выше. Однако вычислительная сложность создания таких наборов на порядок выше даже для небольших пространств поиска и небольших наборов обучающих данных. Поэтому набирающая популярность идея замены обучения модели на предсказание ее качества в данном случае заслуживает особого внимания. Однако ошибка, вносимая использованием такого метода, может быть слишком большой, и тогда оценка, полученная на таком наборе данных, не является репрезентативной.

Интерес к подбору архитектур сетей специально под вычислительное устройство (программно-аппаратную платформу) может добавлять смещение в процесс оценки алгоритма. Это смещение может быть вызвано процедурой оценки производительности или ограничениями в поддержке операций на целевом устройстве. Разработка набора данных для оценки алгоритмов поиска под специализированное устройство сталкивается с проблемами воспроизводимости, так как устройства могут иметь различия как в программном обеспечении, так и в физической реализации логики на кристаллах процессоров.

Заключение

Анализируя методы оценки качества алгоритмов поиска архитектур нейронных сетей, можно заметить, что появляется тенденция на снижение вычислительной сложности оценки качества. Данная идея реализуется через обучение моделей для предсказания качества архитектур. Проблема снижения вычислительной сложности, а вместе с этим и экологичности машинного обучения, поднимается и в статье [6] и является, безусловно, важной. Возможным решением этой проблемы в задаче оценки качества алгоритмов поиска является использование байесовских методов [25, 29, 30] для выбора архитектур или подбора гиперпараметров.

Гиперпараметры обучения моделей во время выбора архитектуры, а также дообучения выбранной модели играют важную роль и привлекают внимание исследователей. Если пространство возможных архитектур может являться конечным и основываться на возможностях аппаратных вычислителей, то задача поиска гиперпараметров имеет континуальное количество вариантов, что часто требует иных подходов. Дискретизация пространства гиперпараметров является одним из решений данной проблемы, однако при этом теряется общность полученного результата, так как при другой дискретизации выводы исследователей могут измениться.

Задачи, решаемые при помощи нейронных сетей, не ограничиваются обработкой изображений и анализом табличных данных, однако для разных направлений часто используются разные архитектуры. Современным требованием в задачах компьютерного зрения является использование архитектуры трансформеров, изначально разработанной для анализа текстов. Применение поиска нейросетевых архитектур к трансформерам может привести к появлению в некоторой мере универсальных моделей, применимых для данных разной природы. В контексте оценки качества таких алгоритмов поиска стоит отметить, что ввиду большей вычислительной сложности самих моделей потребуются методы, использующие ограниченное количество сравнений архитектур для оценки качества алгоритма поиска.

Список литературы / References

1. **Real E., Moore S., Selle A., Saxena S., Suematsu Y. L., Tan J., Le Q. V., Kurakin A.** Large-Scale Evolution of Image Classifiers [Электронный ресурс]. Режим доступа: https://arxiv.org/pdf/1703.01041.pdf (дата обращения: 27.12.22).
2. **Kandasamy K., Neiswanger W., Schneider J., Poczos B., Xing E. P.** Neural Architecture Search with Bayesian Optimisation and Optimal Transport [Электронный ресурс]. Режим доступа: https://proceedings.neurips.cc/paper/2018/file/f33ba15effa5c10e873bf3842afb46a6-Paper.pdf (дата обращения: 27.12.22).
3. **Liu H., Simonyan K., Yang Y.** DARTS: DIFFERENTIABLE ARCHITECTURE SEARCH [Электронный ресурс]. Режим доступа: https://arxiv.org/pdf/1806.09055.pdf (дата обращения: 27.12.22).
4. **Zoph B., Le Q. V.** NEURAL ARCHITECTURE SEARCH WITH REINFORCEMENT LEARNING [Электронный ресурс]. Режим доступа: https://arxiv.org/pdf/1611.01578.pdf (дата обращения: 27.12.22).

5. **Zoph B., Vasudevan V., Shlens J., Le Q. V.** Learning Transferable Architectures for Scalable Image Recognition [Электронный ресурс]. Режим доступа: https://arxiv.org/pdf/1707.07012.pdf%27 (дата обращения: 27.12.22).
6. **Cai H., Gan C., Wang T., Zhang Z., Han S.** ONCE-FOR-ALL: TRAIN ONE NETWORK AND SPECIALIZE IT FOR EFFICIENT DEPLOYMENT [Электронный ресурс]. Режим доступа: https://arxiv.org/pdf/1908.09791.pdf (дата обращения: 27.12.22).
7. **Bichen Wu B., Dai X., Zhang P., Wang Y., Sun F., Wu Y., Tian Y., Vajda P., Jia Y., Keutzer K.** FBNet: Hardware-Aware Efficient ConvNet Designvia Differentiable Neural Architecture Search [Электронный ресурс]. Режим доступа: https://arxiv.org/pdf/1812.03443.pdf (дата обращения: 27.12.22).
8. **Tan M., Le Q. V.** EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks [Электронный ресурс]. Режим доступа: https://arxiv.org/pdf/1905.11946.pdf (дата обращения: 27.12.22).
9. **Zoph B., Vasudevan V., Shlens J., Le Q. V.** Learning Transferable Architectures for Scalable Image Recognition [Электронный ресурс]. Режим доступа: https://arxiv.org/pdf/1707.07012.pdf (дата обращения: 27.12.22).
10. **Krizhevsky A., Sutskever I., Hinton G. E.** ImageNet Classification with Deep Convolutional Neural Networks [Электронный ресурс]. Режим доступа: https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf (дата обращения: 27.12.22).
11. **Hochreiter S., Schmidhuber J.** LONG SHORT-TERM MEMORY [Электронный ресурс]. Режим доступа: http://www.bioinf.jku.at/publications/older/2604.pdf (дата обращения: 27.12.22).
12. **Williams R. J.** Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning [Электронный ресурс]. Режим доступа: https://link.springer.com/content/pdf/10.1007/BF00992696.pdf?pdf=button (дата обращения: 27.12.22).
13. Dataset Website [Электронный ресурс]. Режим доступа: https://www.cs.toronto.edu/~kriz/cifar.html (дата обращения: 27.12.22).
14. **Huang G., Liu Z., van der Maaten L., Weinberger K. Q.** Densely Connected Convolutional Networks [Электронный ресурс]. Режим доступа: https://arxiv.org/pdf/1608.06993.pdf (дата обращения: 27.12.22).
15. **Liu C., Zoph B., Neumann M., Shlens J., Hua W., Li L.-J., Fei-Fei L., Yuille A., Huang J., Murphy K.** Progressive Neural Architecture Search [Электронный ресурс]. Режим доступа: https://download.arxiv.org/pdf/1712.00559v3.pdf (дата обращения: 27.12.22).
16. **Sekanina L.** Neural Architecture Search and Hardware Accelerator Co-Search: A Survey [Электронный ресурс]. Режим доступа: https://ieeexplore.ieee.org/document/9606893 (дата обращения: 27.12.22).
17. **Ying C., Klein A., Real E., Christiansen E., Murphy K., Hutter F.** NAS-Bench-101: Towards Reproducible Neural Architecture Search [Электронный ресурс]. Режим доступа: https://arxiv.org/pdf/1902.09635.pdf (дата обращения: 27.12.22).
18. **Dong X., Yang Y.** NAS-BENCH-201: EXTENDING THE SCOPE OF REPRODUCIBLE NEURAL ARCHITECTURE SEARCH [Электронный ресурс]. Режим доступа: https://arxiv.org/pdf/2001.00326.pdf (дата обращения: 27.12.22).
19. **Zela A., Siems J., Hutter F.** NAS-BENCH-1SHOT1: BENCHMARKING AND DISSECTIN ONE-SHOT NEURAL ARCHITECTURE SEARCH [Электронный ресурс]. Режим доступа: https://arxiv.org/pdf/2001.10422.pdf (дата обращения: 27.12.22).
20. **Li C., Yu Z., Fu Y., Zhang Y., Zhao Y., You H., Yu Q., Wang Y., Lin Y.** HW-NAS-BENCH: HARDWARE-AWARE NEURAL ARCHITECTURE SEARCH BENCHMARK [Электронный ресурс]. Режим доступа: https://arxiv.org/pdf/2103.10584.pdf (дата обращения: 27.12.22).

21. **Klein A., Hutter F.** Tabular Benchmarks for Joint Architecture and Hyperparameter Optimization [Электронный ресурс]. Режим доступа: https://arxiv.org/pdf/1905.04970.pdf (дата обращения: 27.12.22).
22. **Dong X., Yang Y.** Searching for A Robust Neural Architecture in Four GPU Hours [Электронный ресурс]. Режим доступа: https://arxiv.org/pdf/1910.04465.pdf (дата обращения: 27.12.22).
23. **Pham H., Guan M. Y., Zoph B., Le Q. V., Dean J.** Efficient Neural Architecture Search via Parameter Sharing [Электронный ресурс]. Режим доступа: https://arxiv.org/pdf/1802.03268.pdf (дата обращения: 27.12.22).
24. **Kingma D. P., Ba J. L.** ADAM: A METHOD FOR STOCHASTIC OPTIMIZATION [Электронный ресурс]. Режим доступа: https://arxiv.org/pdf/1412.6980.pdf (дата обращения: 27.12.22).
25. **Falkner S., Klein A., Hutter F.** BOHB: Robust and Efficient Hyperparameter Optimization at Scale [Электронный ресурс]. Режим доступа: https://arxiv.org/pdf/1807.01774.pdf (дата обращения: 27.12.22).
26. **Hirose Y., Yoshinari N., Shirakawa S.** NAS-HPO-Bench-II: A Benchmark Dataset on Joint Optimization of Convolutional Neural Network Architecture and Training Hyperparameters [Электронный ресурс]. Режим доступа: https://arxiv.org/pdf/2110.10165.pdf (дата обращения: 27.12.22).
27. **Xu K., Hu W., Leskovec J., Jegelka S.** HOW POWERFUL ARE GRAPH NEURAL NETWORKS? [Электронный ресурс]. Режим доступа: https://arxiv.org/pdf/1810.00826.pdf (дата обращения: 27.12.22).
28. **Yu J., Jin P., Liu H., Bender G., Kindermans P.-L. Tan M., Huang T., Song X., Pang R., Le Q.** BigNAS: Scaling Up Neural Architecture Search with Big Single-Stage Models [Электронный ресурс]. Режим доступа: https://arxiv.org/pdf/2003.11142.pdf (дата обращения: 27.12.22).
29. **Bergstra J., Bardenet R., Bengio Y., Kegl B.** Algorithms for Hyper-Parameter Optimization [Электронный ресурс]. Режим доступа: https://proceedings.neurips.cc/paper/2011/file/86e8f7ab32cfd12577bc2619bc635690-Paper.pdf (дата обращения: 27.12.22).
30. **Li L., Jamieson K., DeSalvo G., Rostamizadeh A., Talwalkar A.** Hyperband: A Novel Bandit-Based Approach to Hyperparameter Optimization [Электронный ресурс]. Режим доступа: https://arxiv.org/pdf/1603.06560.pdf (дата обращения: 27.12.22).

Информация об авторе

Щербин Андрей Сергеевич, аспирант кафедры общей информатики факультета информационных технологий Новосибирского государственного университета
SPIN 3542-1054

Information about the Author

Shcherbin S Andrey, Phd student at Common Informatics Chair of the Information Technologies Department of Novosibirsk State University
SPIN 3542-1054

Статья поступила в редакцию 26.03.2023;
одобрена после рецензирования 30.05.2023; принята к публикации 30.05.2023

The article was submitted 09.12.2022;
approved after reviewing 30.05.2023; accepted for publication 30.05.2023

Правила оформления текста рукописи

Авторы представляют статьи на русском или английском языке объемом от 0,5 авторского листа (20 тыс. знаков) до 1 авторского листа (40 тыс. знаков), включая иллюстрации (1 иллюстрация форматом 190 × 270 мм = 1/6 авторского листа, или 6,7 тыс. знаков). Публикации, превышающие указанный объем, допускаются к рассмотрению только после индивидуального согласования с редакцией журнала.

Текст рукописи должен быть представлен в редколлегию в виде файла MS Word (.doc, .docx). Гарнитура Times New Roman, размер шрифта 11, межстрочный интервал 1, размеры полей – стандартные значения текстового редактора. Форматирование – выравнивание по ширине страницы, переносы слов включены, каждый новый абзац начинается с красной строки. Не допускается ручное форматирование абзацев (пробелами, лишними переводами строк, разрывами страниц).

Структура статьи

- Индекс УДК (универсальной десятичной классификации). Выравнивание по левому краю
- Название статьи. Выравнивание по центру, полужирный шрифт
- ФИО авторов (полностью). Выравнивание по центру, полужирный шрифт
- Места работы всех авторов. Выравнивание по центру, курсив
- Адреса электронной почты, ORCID авторов
- Аннотация статьи
- Ключевые слова, не более 10
- Благодарности, сведения о финансовой поддержке
- Название статьи **на английском языке**. Выравнивание по центру, полужирный шрифт
- ФИО авторов **на английском языке** (полностью). Выравнивание по центру, полужирный шрифт
- Места работы авторов **на английском язык**е. Выравнивание по центру, курсив
- Аннотация статьи **на английском языке (Abstract)**, 200–250 слов
- Ключевые слова **на английском языке (Keywords)**, не более 10
- Благодарности, сведения о финансовой поддержке **на английском язык**е, если есть соответствующий раздел на русском языке **(Acknowledgements)**
- Основной текст
- Список литературы / **References**
- Сведения об авторах

Требования к оформлению основного текста и иллюстративных материалов

Основной текст должен быть представлен в структурированном виде, рекомендуется использовать подзаголовки – например: Введение, Методика..., Выводы, Результаты, Заключение.

Подзаголовки отделяются и набираются полужирным шрифтом. В целях выделения частей текста и отдельных слов и словосочетаний допускается использование курсива или полужирного шрифта. Подчеркивание, разрядка, изменение основного кегля и выделение цветом не используются.

Иллюстрации к рукописи статьи должны быть приложены в виде отдельных файлов. При этом в тексте должно содержаться включенное изображение с указанием имени файла. Все иллюстрации, содержащие схемы, графики, алгоритмы и т. п., должны быть представлены в векторном виде (.ai, .eps, .cdr). Скриншоты и другие растровые изображения должны быть представлены в максимально высоком качестве, без каких-либо потерь и искажений (.jpg, .tif). Все иллюстрации должны иметь подрисуночную подпись – свое название. Надписи к таблицам и подписи к иллюстрациям приводятся **на двух языках (русском и английском)**.

Примеры:

Рис. 1. Диаграмма производительности...
Fig. 1. Performance diagram...

Таблица 1

Сравнение алгоритмов...

Table 1

Comparison of algorithms...

Нумерация последовательная и неразрывная от начала статьи. Не допускается использование других наименований, кроме «Рис.» / «Fig.», «Таблица» / «Table», и усложнение нумерации (например, «Рис. 3.2.»). Ссылка на иллюстрацию в тексте должна быть приведена в круглых скобках, например: (рис. 1), (табл. 1).

Формулы должны быть набраны с использованием редактора MathType либо встроенного редактора формул MS Word. Кегль основных символов – 11, греческие символы набираются прямым шрифтом, латинские – курсивом. Нумеруются только те формулы, на которые автор ссылается в тексте.

Abstract

Аннотация статьи на английском языке (Abstract) не должна быть дословным переводом русскоязычной аннотации. Раздел Abstract, как и основной текст, должен быть структурирован, в нем должно содержаться описание цели работы, методов исследования, научной значимости, выводов / результатов. Требуется качественный перевод на английский язык (при необходимости просим авторов обращаться к профессиональным переводчикам). **Объем Abstract 200–250 слов.**

Список литературы / References

Список литературы и список литературы на английском языке (References) размещаются в общем разделе. Рекомендуемое количество цитируемых в статье источников – не менее 10, в список желательно включать ссылки на актуальные работы по теме исследования, особенно в иностранных периодических изданиях.

В тексте статьи ссылки на литературу указываются цифрами в квадратных скобках, при необходимости указываются номера страниц, например: [2; 3. С. 15].

Список литературы нумеруется в порядке цитирования и оформляется в соответствии с ГОСТ Р 7.0.5-2008 на библиографическое описание (знаки тире в описании опускаются). Ссылки на неопубликованные работы, а также на Интернет-ресурсы (кроме электронных изданий, поддающихся библиографическому описанию) оформляются в виде сноски.

В Список литературы ссылки на источники следует включать на оригинальном языке опубликования. Каждый источник должен быть также оформлен на английском языке (References) по международному стандарту для публикаций в области информатики IEEE Style со следующими отличиями:

- инициалы авторов указываются после фамилии;
- название статьи не берется в кавычки, отделяется точкой;

- отсутствует союз «and» перед фамилией последнего автора;
- в диапазоне страниц – удвоенная «p» (например, «pp. 2–9»);
- год издания указывается после места издания (для книг) и сразу после названия журнала (для периодики).
- Перевод источника на английский язык:
- если источник имеет выходные данные на английском языке, то для формирования References **следует использовать именно эти данные**;
- если оригинальная публикация не содержит выходных данных на английском языке, то допускается транслитерация названия материала на латинский алфавит в сочетании с переводом на английский язык в квадратных скобках. В конце описания указывается, на каком языке написана эта работа, например, (in Russ.). При транслитерации можно воспользоваться Интернет-ресурсом http://ru.translit.ru/, рекомендуется выбрать стандарт BSI. Место издания не транслитерируется, указывается полностью на английском языке, например: Moscow. Название издательства / издателя, как правило, транслитерируется. Для журналов, у которых есть официальное название на английском языке, – использовать его (проверить на сайте журнала, или, например, в библиотеке WorldCat), если названия на английском языке нет, использовать транслитерацию по системе BSI. Не следует самостоятельно переводить названия журналов.

Если у цитируемого источника есть **цифровой идентификатор DOI** (https://search. crossref. org), его требуется обязательно указывать в конце библиографической ссылки.

Примеры оформления ссылок. Каждый источник в том же пункте дублируется на английском языке (References).

Источник на русском языке, перевод на английский доступен в метаданных статьи

1. **Журавлев С. С., Рудометов С. В., Окольнишников В. В., Шакиров С. Р.** Применение модельно-ориентированного проектирования к созданию АСУ ТП опасных промышленных объектов // Вестник НГУ. Серия: Информационные технологии. 2018. Т. 16, № 4. С. 56–67. DOI 10.25205/1818-7900-2018-16-4-56-67

Zhuravlev S. S., Rudometov S. V., Okolnishnikov V. V., Shakirov S. R. Model-Based Design Approach for Development Process Control Systems of Hazardous Industrial Facilities. *Vestnik NSU. Series: Information Technologies*, 2018, vol. 16, no. 4, pp. 56–67. (in Russ.) DOI 10.25205/ 1818-7900-2018-16-4-56-67

Источник на английском языке. Оформляем согласно требованиям для References. Приводим только 1 раз.

2. **Telnov V. I.** Optimization of the Beam Crossing Angle at the ILC for E + e- and yy Collisions. *Journal of Instrumentation*, 2018, vol. 13, no. 03, pp. P03020–P03020. DOI 10.1088/1748-0221/13/03/ p03020

Метаданные источника доступны только на русском языке

3. **Жижимов О. Л., Федотов А. М., Шокин Ю. И.** Технологическая платформа массовой интеграции гетерогенных данных // Вестник НГУ. Серия: Информационные технологии. 2013. Т. 11, вып. 1. С. 24–41.

Zhizhimov O. L., Fedotov A. M., Shokin Yu. I. Tekhnologicheskaya platforma massovoi integratsii geterogennykh dannykh [Technology Platform for the Mass Integration of Heterogeneous Data]. *Vestnik NSU. Series: Information Technologies*, 2013, vol. 11, no. 1, pp. 24–41. (in Russ.)

Сведения об авторах

Последний раздел статьи – информация об авторе / авторах **на русском и английском языках**:

- ФИО полностью, ученая степень, ученое звание;
- идентификаторы автора, такие как ResearcherID (всем авторам рекомендуется использовать данные сервисы для ведения актуального списка своих публикаций);
- контактный телефон (не публикуется).

Если статья представляется на английском языке, необходимо приложить перевод на русский язык названия, аннотации, ключевых слов, сведений об авторе.

Доставка материалов

Материалы предоставляются в редакцию по электронной почте inftech@vestnik.nsu.ru.

Порядок рецензирования

Все статьи сначала проходят проверку на заимствование и только после этого отправляются на рецензирование. Редакционный совет не допускает к публикации материал, если имеется достаточно оснований полагать, что он является плагиатом.

Тип рецензирования статей – двухуровневое, одностороннее анонимное («слепое»).

Для каждой статьи редколлегией выбираются рецензенты, научная деятельность которых связана с темой представленного материала. Ответственный секретарь журнала обращается к ним с просьбой дать экспертную оценку статье либо помочь организовать рецензирование.

Рецензии для журнала «Вестник НГУ. Серия: Информационные технологии» составляются по единой схеме и подразумевают оценку по следующим критериям: соответствие тематике журнала, оригинальность и значимость результатов, качество изложения материала.

Заполненный бланк рецензии высылается на электронный адрес редакции. В зависимости от экспертных заключений статья может быть принята редакционным советом к опубликованию, рекомендована автору к доработке (с последующим повторным рецензированием либо без него) или отклонена (с предоставлением автору мотивированного отказа). Автору на электронный адрес высылается текст рецензии без указания ФИО рецензента и его контактных данных.

Все рецензии хранятся в редакции журнала не менее 5 лет. Редколлегия журнала обязуется при поступлении соответствующего запроса направлять копии рецензий в Министерство науки и высшего образования Российской Федерации.