Московский Государственный Университет
имени М.В. Ломоносова
Российская Академия Наук
Международная Академия Технологических Наук
Российская Академия Естественных Наук

Интеллектуальные Системы.

Теория и приложения

ТОМ **26** ВЫПУСК 4 * **2022**

МОСКВА

ISSN 2411–4448

УДК 519.95; 007:159.955
ББК 32.81

Издается с 1996 г.*

В журнале «Интеллектуальные системы. Теория и приложения» публикуются научные достижения в области теории и приложений интеллектуальных систем, новых информационных технологий и компьютерных наук.

Издание журнала осуществляется под эгидой МГУ имени М. В. Ломоносова, Научного Совета по комплексной проблеме «Кибернетика» РАН, Отделения «Математическое моделирование технологических процессов» МАТН, Секции «Информатики и кибернетики» РАЕН.

Учредитель журнала: ООО «Интеллектуальные системы».

Журнал входит в список изданий, включенных ВАК РФ в реестр публикаций материалов по кандидатским и докторским диссертациям по математике и механике.

Индекс подписки на журнал: 64559 в каталоге НТИ «Роспечать».

Адрес редакции: 119991, Москва, ГСП-1, Ленинские Горы, д. 1, механико-математический факультет, комн. 12-01.
Адрес издателя: 115230, Россия, Москва, Хлебозаводский проезд, д. 7, стр. 9, офис 9. Тел. +7 (495) 939-46-37, e-mail: mail@intsysjournal.org

*) Прежнее название журнала: «Интеллектуальные системы».

ОГЛАВЛЕНИЕ

Часть 1.
Общие проблемы теории интеллектуальных систем

Возможности для реализации системы автоматного обучения

В. А. Бирюкова[1], Г. В. Боков[2], А. С. Дробышев[3], Г. В. Калачев[4], В. С. Половников[5], Д. В. Ронжин[6], А. А. Часовских[7]

[1] *Бирюкова Вероника Андреевна* — аспирант каф. математической теории интеллектуальных систем мех.-мат. ф-та МГУ, e-mail: biryukovaveronika@mail.ru

Biryukova Veronika Andreevna — Postgraduate, Lomonosov Moscow State University, Faculty of Mechanics and Mathematics, Chair of Mathematical Theory of Intellectual Systems.

[2] *Боков Григорий Владимирович* — кандидат физико-математических наук, доцент каф. математической теории интеллектуальных систем мех.-мат. ф-та МГУ, e-mail: bokov@intsys.msu.ru

Bokov Grigoriy Vladimirovich — Candidate of Physical and Mathematical Science, Associate Professor, Lomonosov Moscow State University, Faculty of Mechanics and Mathematics, Chair of Mathematical Theory of Intellectual Systems.

[3] *Дробышев Александр Сергеевич* — аспирант каф. математической теории интеллектуальных систем мех.-мат. ф-та МГУ, e-mail: drobyshev.sanya@yandex.ru

Drobyshev Alexander Sergeevich — Postgraduate, Lomonosov Moscow State University, Faculty of Mechanics and Mathematics, Chair of Mathematical Theory of Intellectual Systems.

[4] *Калачев Глеб Вячеславович* — кандидат физико-математических наук, младший научный сотрудник каф. математической теории интеллектуальных систем мех.-мат. ф-та МГУ, e-mail: gleb.kalachev@yandex.ru

Kalachev Gleb Vyacheslavovich — Candidate of Physical and Mathematical Science, Junior Researcher, Lomonosov Moscow State University, Faculty of Mechanics and Mathematics, Chair of Mathematical Theory of Intellectual Systems.

[5] *Половников Владимир Сергеевич* — кандидат физико-математических наук, научный сотрудник каф. математической теории интеллектуальных систем мех.-мат. ф-та МГУ, e-mail: pvser@mail.ru

Polovnikov Vladimir Sergeevich — Candidate of Physical and Mathematical Science, Researcher, Lomonosov Moscow State University, Faculty of Mechanics and Mathematics, Chair of Mathematical Theory of Intellectual Systems.

[6] *Ронжин Дмитрий Владимирович* — кандидат физико-математических наук, младший научный сотрудник каф. математической теории интеллектуальных систем мех.-мат. ф-та МГУ, e-mail: d_rongin@mail.ru

Ronzhin Dmitry Vladimirovich — Candidate of Physical and Mathematical Science, Junior Researcher, Lomonosov Moscow State University, Faculty of Mechanics and Mathematics, Chair of Mathematical Theory of Intellectual Systems.

[7] *Часовских Анатолий Александрович* — доктор физико-математических наук доцент каф. математической теории интеллектуальных систем мех.-мат. ф-та МГУ, e-mail: chasovskikh@mail.ru.

Chasovskikh Anatoly Alexandrovich — Doctor of Physical and Mathematical Sciences, Associate Professor, Lomonosov Moscow State University, Faculty of Mechanics and Mathematics, Chair of Mathematical Theory of Intellectual Systems.

Приведены результаты анализа некоторых доступных источников, на основе которого была реализована автоматная модель обучения искусственных нейронных сетей.

Ключевые слова: распознавание изображений, искусственная нейронная сеть, автоматизированное машинное обучение, автоматная модель обучения, гиперпараметры.

В этой заметке приведены результаты анализа доступных источников, на основе которого была реализована автоматная модель обучения искусственных нейронных сетей, представленная в работе [1].

1. Введение.

Для эффективного обучения искусственных нейронных сетей (ИНС) следует решить ряд задач: подобрать данные для обучения и тестирования, определить архитектуру ИНС, выполнить предобработку данных, задать метрики эффективности получаемых нейросетевых решений, выбрать параметры обучения. Только после этого обучение ИНС может быть начато. Неудовлетворительные результаты тестирования обученной модели ИНС приводят к изменению начальных установок, перечисленных выше, а затем, к возобновлению процесса обучения. Количество подходов к обучению ИНС зависит, естественно, от ряда причин, но, как правило, процесс обучения занимает значительное время у разработчика, особенно у не имеющего соответствующего опыта.

Возможность обучения ИНС распознаванию образов во многом предопределяется наличием в открытом доступе баз данных (БД). Эти базы, если и не позволяют перейти к обучению ИНС поставленной прикладной задаче, то, как правило, помогают получить предварительно обученную сеть, способную выделять необходимые признаки из обрабатываемых данных. Второй этап обучения предварительно обученной ИНС выполняется, как правило, на собранной пользователем специализированной БД. Мы приведем здесь краткое описание некоторых известных БД для анализа изображений и реализованный нами подход к унификации разметок, используемых в разных БД.

Для оптимизации процесса обучения ИНС используются методы AutoML, получившие в настоящее время широкое распространение. В этой работе мы приводим некоторые из этих методов, рассмотренных в [2].

Задание правильных гиперпараметров ИНС и алгоритма ее обучения является важнейшим условием получения удовлетворительного обучения модели, редко получается у не искушенного разработчика. Естественным желанием поэтому является использование автоматического

задания гиперпараметров. В этой работе приведены некоторые подходы к решению этой задачи. Для реализации собственной системы, как уже было сказано в [1], мы использовали градиентный метод поиска по сетке и метод «лучших практик».

2. Некоторые известные базы изображений и унификация разметки.

Разработанная нами система автоматного обучения ИНС распознаванию визуальных образов использует накопленные массивы изображений, примеры которых мы приведем далее. Начнем с наиболее часто встречающихся в статьях, являющихся эталонными, универсальными:

- ImageNet [3] Используется для решения задачи классификации объектов. Содержит по 1300 изображений объектов из 1000 классов. Свободно распространяются сети, прошедшие предварительное обучение на этой БД. Ввиду большого количества классов, обучение на ImageNet позволяет сети выделять значительный объем признаков.

- PascalVOC [4] Содержит изображения объектов из 20 классов. Изображения размечены для решения задач детектирования и сегментации, для сегментации разметка достаточно.

- COCO [5] Универсальная база данных для решения задач распознавания визуальных образов объектов из 80 классов. Каждое изображение содержит, как правило, несколько объектов. Изображения размечены для решения задач классификации, детектирования, сегментации, поиска ключевых точек. Далее приведем примеры БД изображений, специализированных по тематике решаемой задачи или ориентированной на определенный устоявшийся алгоритм решения некоторой задачи:

- Kaggle Kat and Dog images for Classification Содержит 25000 изображений кошек и собак для решения задачи бинарной классификации заданных классов объектов.

- KITTI [6] База данных сегментации вида из автомобиля, построения карты глубины и детектирования машин двумерными и трехмерными охватывающими прямоугольниками и параллелепипедами соответственно. База может использоваться для обучения системы управления самодвижущим автомобилем.

- [7] Содержит изображения городских ландшафтов и карты их сегментации. Может использоваться для решения задач интеллектуального управления транспортным потоком.

- FDDB: Face Detection Data Set and Benchmark [8], Wider Face [9] Эти базы снимков, содержащие изображения, которые используются для обучения ИНС обнаруживать лица. Вторая из них включает 32 203 изображения, которые содержат 393 703 фрагментов с лицами.

- Menpo Dataset [10] База данных с изображениями лиц с возможностью выделения на лицах ключевых точек. Эта БД может использоваться для решения задачи выравнивания изображения лица, как предварительного этапа их распознавания.

  Наконец, приведем пример базы, используемой для решения специализированных задач детектирования и классификации.

- Russian Traffic Sign [11] Содержит 179138 размеченных снимков, 156 классов объектов, 15630 изображений объектов (для классификации), 104358 включений объектов (для детектирования).

Для реализации системы автоматного обучения нами были отобраны 3 БД: ImageNet, COCO и Kaggle Kat and Dog images for Classification.

ИНС принимает на вход изображение, выдаёт результат в зависимости от задачи 1. Классификация – номер класса объекта, изображенного на данном снимке 2. Сегментация – маску, то есть изображение того же разрешения, что и входное, с разделением на классы или сущности в зависимости от вида сегментации. 3. Детектирование – перечень охватывающих прямоугольников или многоугольников с номером класса для каждого из них. 4. Ключевые точки – последовательность координат для N ключевых точек для каждого вхождения объекта.

Следует отметить, что существуют архитектуры, позволяющие решать сразу несколько из перечисленных задач. К ним относится, например, Mask RCNN, которая может быть обучена для решения сразу всех четырех перечисленных задач.

Для обучения и тестирования ИНС требуются файлы разметки (аннотации), и они могут быть в различных форматах. Поэтому в разработанной нами системе эти файлы приводятся к единому формату, используемому в COCO.

3. Некоторые методы мета-обучения.

Ряд публикаций посвящен обзору существующих систем AutoML и описанию подходов для создания подобных систем. Для обзора нами использована работа [2].

С математической точки зрения всякая система AutoML может быть формализована следующим образом: Пусть T – множество всех задач, которые потенциально могут быть решены при помощи инструментов машинного обучения. Пусть Θ – множество всех конфигураций для всех алгоритмов машинного обучения, причем Θ описывает пространство параметров произвольной структуры такое, что всякий алгоритм машинного обучения полностью характеризуется некоторым элементов из Θ (в т.ч. гиперпараметры алгоритма – количество слоев нейронной сети, размеры обучающей выборки, параметры оптимизационных алгоритмов и т.п.). Пусть также имеется множество P всех скалярных результатов измерения эффективности решения некоторой задачи из множества T алгоритмом, описанным параметрами элемента из Θ, т.е.$P_{(i,j)} = P(\theta_i, t_j)$, $\theta_i \in \Theta$, $t_j \in T$. Скалярные значения определяются в соответствии с установленной метрикой и по определенной процедуре измерения, которые определяются заранее. Пусть задано множество измерений производительности P_{new} для некоторой новой задачи t_{new} для некоторых алгоритмов машинного обучения: $P_{new} = P_{\{}(i, new)\}$. Система AutoML, по существу, представляет собой некоторую модель L которая осуществляет рекомендацию для новой задачи t_{new} некоторого оптимального набора параметров Θ^* исходя из данных, полученных по множеству P_{new}.

Особый интерес представляют рекомендательные системы, которые не зависят от задачи. В случае подобных систем можно считать $P_{new} = \emptyset$, и в ходе создания рекомендательной системы для такого случая обучается функция, которая выдает набор предпочтительных конфигураций $f : \Theta \times T \to \theta_k^*$, $k = 1 \dots K$, причем среди $\{\theta_k^*\}$ выбирается лучший алгоритм для использования или дальнейшей оптимизации. Множество $\{\theta_k^*\}$ может быть отранжировано не только по качеству, но и по скорости обучения.

При решении задач создания AutoML систем отдельным образом выделяются Байесовские подходы. Так, для обучения f во многих работах используется моделирование Гауссовским процессом, либо совместно для всех задач, либо для каждой задачи отдельно (так называемые суррогатные модели), с дальнейшей комбинацией задач (например, при помощи взвешенной нормализованной суммы).

Подходы, связанные с обучением Гауссовских процессов для создания рекомендательных систем при подборе гиперпараметров, несмотря на их универсальность, отличаются сложностью обучения и большими

вычислительными затратами. В силу данного обстоятельства нередко производится предварительная оптимизация пространства поиска гиперпараметров с целью ускорения обучения модели для рекомендаций. Так, к примеру, для ускорения работы можно ограничивать размерность пространства Θ:

- Выбирая некоторое число атрибутов, которые дают наибольший разброс на заданной задаче, либо на всех задачах совместно
- Выбирая некоторое число атрибутов, которые дают наибольший прирост качества (по мере оценивания), при фиксированных прочих атрибутах
- Выбирая некоторое число атрибутов, которые дают наименьшее ухудшение качества (по мере оценивания), при фиксированных прочих атрибутах.

В качестве широко используемых компонент, применяемых в системах AutoML могут быть упомянуты следующие.

- Active Testing.

  Данный подход использует метрики для оценки близости конфигураций, на основании чего в ходе поиска оптимальной конфигурации для задачи выделяются ближайшие к заданной известные задачи. В качестве примера может быть рассмотрена метрика relative landmarks: $RL_{(a,b,j)} = P_{(a,j)} - P_{(b,j)}$, для конфигураций θ_a, θ_b и задачи t_j. Идея данного подхода может быть описана следующей циклической процедурой:
  1) выбираем некоторую «лучшую» конфигурацию,
  2) выбираем «лучшую» конфигурацию на «близких» задачах по мере RL,
  3) обновляем RL,
  4) повторяем 1)-3) до удовлетворения некоторому критерию остановки.

- Суррогатные модели.

  Данный подход заключается в обучении модели для каждой известной ранее задачи, на основании проделанных ранее замеров эффективности: строятся модели $s_j(\theta_i) = P_{(i,j)}$ для всех t_j используя P. Далее, после обучения суррогатных моделей для всякой новой задачи выбирается ансамбль моделей, которые обучены для близких к данной задачах. Близость задач может оцениваться с использованием некоторой метрики, как упомянуто выше. Однако, можно

использовать и активный подход для сравнения. Например, задачи можно считать близкими, если суррогатная модель задачи t_j генерирует близкие предсказания для задачи t_new.

- Использование кривых обучения.

  В случае, если накоплена информация о процессе обучения различных моделей, можно использовать эти данные для построения предсказательных систем. Например, обучение разделяется на шаги r_t (например, добавляя одинаковое количество новых данных для обучения), и вычисляются значения $P(\theta_i, t_j, r_t) = P_{(i,j,t)}$, тем самым задается кривую обучения алгоритма θ_i на задаче t_j. Проводя оценку задачи t_new на алгоритме θ_i можно останавливать обучение на некотором шаге, и (возможно учитывая другие оценки близости t_{new} и остальных t_i) делать предсказание по частично построенной кривой, насколько хорошо данный алгоритм отработает на полной базе данных. Таким образом поиск потенциально хороших конфигураций может быть ускорен.

- Выбор конфигурации по ближайшим соседям.

  При заданной мере близости задач, выбор конфигурации может быть сведен к выбору k ближайших задач (алгоритмом kNN), и, соответственно, k наилучших конфигураций для этих задач, например при помощи суррогатных моделей.

- Использование мета-атрибутов.

  Различные задачи (частично характеризуемые базами данных, на которых данные задачи обучаются и тестируются) обладают определенными характерными параметрами, которые называются мета-атрибутами. Примерами таких атрибутов могут служить статистические характеристики баз данных, такие как среднее значение и разброс определенной характеристики, объемы баз данных, количество пропущенных атрибутов. Одним из подходов при обучении AutoML систем является определение близости задач через близость мета-атрибутов по некоторой метрике, для выбора наилучших конфигураций среди близких задач.

  Следует отметить, что при вычислении мета-атрибутов обязательна их нормализация, а иногда требуется уменьшение размерности задачи.

- Обучение мета-атрибутам

  Вместо непосредственного вычисления мета-атрибутов для задачи возможен вариант обучения модели, для построения новых мета-атрибутов. Идея подхода заключается в следующем. Генерируется

набор бинарных мета-атрибутов $m_{(j,a,b)}$, который показывает будет ли задача t_j лучше решена при помощи θ_a чем θ_b. Для создания таких атрибутов обучается модель, принимающая пару (a, b) и задачу t_j, для обучения используется набор P и множество мета-атрибутов, которые могут быть вычислены вручную.

- Предсказание оценки по задаче и конфигурации.

  При использовании данного подхода строится регрессия (линейная, SVM) и по наборам мета-атрибутов и конфигурации θ_i предсказывается для задачи её оценки на конфигурации θ_i, в т.ч. скорость обучения. Результаты предсказания используются для поиска хороших конфигураций в дальнейшей оптимизации.

- Рекомендации по последовательному использованию конфигураций (pipeline).

  Бывает возможным использование модели обучения с подкреплением (reinforcement-learning) для предсказания какие конфигурации использовать после применения выбранной ранее конфигурации, для дальнейшего улучшения качества.

  В качестве примеров систем AutoML, реализованных в виде программных библиотек, которые используются на практике могут быть упомянуты следующие системы: Auto-SKLearn [2] и AutoKeras [12].

4. Методы оптимизации подбора гиперпараметров.

В работе [13] «Практическое руководство по классификации опорных векторов» для получения оптимального набора гиперпараметров предлагается выполнить перебор по решетке их значений. Этот метод высоко затратен по времени и другим ресурсам, но легко параллелизуемый.

Имплементация есть, например, в библиотеке sklearn (scikit-learn) [14] языка программирования Python, в библиотеке Keras этого же языка для этого подхода есть методы или инструменты.

С целью сокращения времени оптимизации гиперпараметров может быть использован метод случайного поиска по сетке их значений [15]. Метод является менее надежным, чем полный перебор, но может быть результативным при использовать дополнительных знаний о распределении возможных значений гиперпараметров или если задача оптимизации имеет низкую внутреннюю размерность.

Известны подходы, связанные с Байесовской оптимизацией. Для этого на первом этапе изучается поведение функции, оценивающей качество распознавания, значения которой определяется на ряде наборов значений гиперпараметров. Затем отбрасываются заведомо плохие их комбинации и выполняется обучение на наиболее перспективных их наборах. Метод при равных ограничениях по времени, как правило, позволяет получить лучшие результаты, чем перебор по решетке и случайный поиск.

Сюда же можно отнести алгоритмы, основанные на обучении с подкреплением (reinforcement learning, RL). Исследование метода содержится в [16]. В библиотеке github можно найти реализацию метода.

Следует отметить градиентные методы. Их использование связано с конкретным алгоритмом обучения, для которого можно вычислить градиенты гиперпараметров и воспользоваться методом градиентного спуска [17]. В своих исследованиях В.С. Половников с учениками использовал оптимизацию гиперпараметра dropout, включив дополнительный параметр в функцию потерь и оптимизировав его вместе с остальными весами.

Градиентный метод, изложенный в статье [18], позволяет оптимизировать следующие гиперпараметры: размер пакета, расписание скорости обучения, распределение весов при инициализации, схемы регуляризации, данные для обучения, архитектуры нейронных сетей. Для этого используется алгоритм стохастического градиентного спуска (stochastic gradient descent, SGD) со слагаемым, содержащим параметр момента, затем находятся формулы для вычисления градиентов. Для получения точных оценок требуется работать с гессианами, но, используя методы оптимизации, удается сократить время их подсчета. Таким образом удается оптимизировать скорость обучения и величину момента. Оптимизация размера пакета осуществляется по методу Hamiltonian Monte Carlo с использованием статьи [19].

Веса нейронной сети, согласно рассматриваемому алгоритму, инициализируются случайно, а оптимизируются вариации этих весов, аналогично подходу, использованному в работе [20]. Регуляризация сводится к взвешенной добавке L2, различным параметрам соответствует различные коэффициенты. Авторы показывают, что, как минимум, для логистической регрессии на MNIST это осмысленно. Предложен алгоритм для оптимизации данных. Путем объединения градиентов посредством преобразований данных авторы могут вычислять градиенты цели валидации в отношении процедур предварительной обработки, взвешивания или увеличения данных.

Оптимизация архитектуры нейронной сети используется в методе из статьи [36], который через множители включает те или иные ветви. В процесс обучения чередуется изменение параметров и этих множителей,

а потом по порогу убираются ветви с низкими коэффициентами. В зависимости от набора удаленных ветвей получаются различные архитектуры нейронных сетей. Метод хорош, но сильно ограничен по пространству поиска (последовательная блочная структура).

Перспективным представляется подход, автором которого является Roger Grosse, связанный с одновременной оптимизации параметров и гиперпараметров. Этому подходу посвящены статьи [21]– [23] и другие исследования.

Следует упомянуть также эволюционные методы, в частности, генетические алгоритмы, основанные на понятии популяции. Оптимизация здесь осуществляется по схеме: вносится мутация (в данном случае, меняются гиперпараметры алгоритма обучения) и после анализа результатов обучения с новыми гиперпараметрами, сравниваем, если новый вариант нейронной сети работает лучше, то дальше развиваем его дальше. Отметим методы, основанные на раннем останове. При этом, например, осуществляется параллельный поиск по сетке, но периодически отбрасываются самые отстающие наборы значений гиперпараметров, а также методы, основанные на схеме моделирования отжига simulated annealing) [24].

Для реализации системы автоматного обучения мы использовали поиск по решетке значений выбранных гиперпараметров, но не полный перебор их значений, а, используя выбор наиболее «перспективных» направлений перемещения по ней.

Исследование выполнено при поддержке Министерства науки и высшего образования Российской Федерации (грант № 075-15-2020-801)

Список литературы

[1] Бирюкова В.А., “Автоматный подход для оптимизации работы системы обучения нейронных сетей”, *Интеллектуальные системы. Теория и приложения*, **25**:4 (2021), 71–78.

[2] Hutter F., Kotthoff L., Vanschoren J., *Automated Machine Learning*, «Springer», Friburg, Germany, 2019, 223 с.

[3] Deng J., Dong W., Socher R., Li L.-J., Li K. and Fei-Fei L., “ImageNet: A Large-Scale Hierarchical Image Database”, *IEEE Conference on Computer Vision and Pattern Recognition*, 2009, 248–255

[4] Everingham, M., Eslami, S. M. A., Van Gool, L., Williams, C. K. I., Winn, J. and Zisserman, A., “The PASCAL Visual Object Classes Challenge: A Retrospective”, *International Journal of Computer Vision*, **111**:1 (2015), 98–136

[5] Lin T.-Y. , Maire M., Belongie S., Bourdev L., Ross G., James H., Perona P., Ramanan D., Zitnick C. L., Dollár P., "Microsoft COCO: Common Objects in Context", *ArXiv,* ***abs/ 1405.0312***, 2014

[6] Geiger A., Lenz P., Stiller C. and Urtasun R., "Vision meets Robotics: The KITTI Dataset", *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012

[7] Cordts M., Omran M., Ramos S., Rehfeld T., Enzweiler M., Benenson R., Franke U., Roth S., Schiele B., "The Cityscapes Dataset for Semantic Urban Scene Understanding", *ArXiv,* ***abs/ 1604.01685***, 2014

[8] Jain V. and Learned-Miller E., *FDDB: A Benchmark for Face Detection in Unconstrained Settings*, University of Massachusetts, Amherst, 2014, 11 c.

[9] Yang, Shuo and Luo, Ping and Loy, Chen Change and Tang, Xiaoou, "WIDER FACE: A Face Detection Benchmark", *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016

[10] Deng J., and Roussos A., Chrysos G., Ververas E., Kotsia I., Shen J., Zafeiriou S., "The menpo benchmark for multi-pose 2d and 3d facial landmark localisation and tracking", *International Journal of Computer Vision*, **127**:6 (2019), 599–624

[11] Шахуро В.И., Конушин А.С., "Российская база изображений автодорожных знаков", *Компьютерная оптика*, **40**:2 (2016), 294–300

[12] Jin Haifeng and Song Qingquan and Hu, Xia, "An Efficient Neural Architecture Search System", *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2019, 1946–1956

[13] Chin-Wei Hsu, Chih-Chung Chang and Chih-Jen Lin, *A practical guide to support vector classification*, National Taiwan University, Taipei, 2010, 16 c.

[14] Aurelien Geron, *Hands-On Mashine Learning with Scikit-Learn, Keras, and TensorFlow, 2nd Edition*, O'Reilly Media, Inc., Sebastopol, California, 2019, 856 c.

[15] James Bergstra, Yoshua Bengio, "Random Search for Hyper-Parameter Optimization", *Machine Learning Research*, **13** (2012), 281–305

[16] Bergstra, J., Yamins, D., Cox, D. D., "Making a Science of Model Search: Hyperparameter Optimization in Hundreds of Dimensions for Vision

Architectures.", *TProc. of the 30th International Conference on Machine Learning (ICML 2013)*, 2013

[17] Larsen J., Hansen L.K., Svarer C., Ohlson B.O.M., "Design and regularization of neural networks: the optimal use of a validation set", *IEEE Signal Processing Society Workshop*, 1996, 62–71

[18] Maclaurin D., Duvenaud D., Adams R. P., "Gradient-based Hyperparameter Optimization through Reversible Learning", *ArXiv,* ***abs/ 1502.03492***, 2015

[19] Salimans T., Kingma D.P., Welling M., "Markov chain Monte Carlo and variational inference: Bridging the gap", *ArXiv,* ***abs/ 1410.6460***, 2014

[20] Chen Zhu, Renkun Ni, Zheng Xu, Kezhi Kong, W. Ronny Huang, Tom Goldstein, "GradInit: Learning to Initialize Neural Networks for Stable and Efficient Training", *ArXiv,* ***abs/ 2102.08098***, 2021

[21] MacKay M., Vicol P., Lorraine J., Duvenaud D., Grosse R., "Self-Tuning Networks: Bilevel Optimization of Hyperparameters using Structured Best-Response Functions", *ArXiv,* ***abs/ 1903.03088***, 2019

[22] Bae J., Grosse R., "Delta-STN: Efficient Bilevel Optimization for Neural Networks using Structured Response Jacobians", *ArXiv,* ***abs/ 2010.13514***, 2020

[23] Franceshi L., Frasconi P., Salso S., Grazzi R., Pontil M., "Bilevel Programming for Hyperparameter Optimization and Meta-Learning", *ArXiv,* ***abs/ 1806.04910***, 2018

[24] Correia A. H.C., Worrall D. E., Bondesan R., "Neural Simulated Annealing", *ArXiv,* ***abs/ 2203.02201***, 2022

Opportunities for the implementation of automatic learning system

Biryukova V.A., Bokov G.V., Drobyshev A.S., Kalachev G.V., Polovnikov V.S., Ronzhin D.V., Chasovskikh A.A.

The results of the analysis of some available sources are presented, on the basis of which an automatic learning model for artificial neural networks was implemented.

Keywords: image recognition, artificial neural network, automated machine learning, automatic learning model, hyperparameters.

References

[1] Biryukova V.A., "Optimization of neural network learning via system with automata approach", *Intelligent Systems. Theory and Applications*, **25**:4 (2021), 71–78 (In Russian)

[2] Hutter F., Kotthoff L., Vanschoren J., *Automated Machine Learning*, «Springer», Friburg, Germany, 2019, 223 c.

[3] Deng J., Dong W., Socher R., Li L.-J., Li K. and Fei-Fei L., "ImageNet: A Large-Scale Hierarchical Image Database", *IEEE Conference on Computer Vision and Pattern Recognition*, 2009, 248–255

[4] Everingham, M., Eslami, S. M. A., Van Gool, L., Williams, C. K. I., Winn, J. and Zisserman, A., "The PASCAL Visual Object Classes Challenge: A Retrospective", *International Journal of Computer Vision*, **111**:1 (2015), 98–136

[5] Lin T.-Y. , Maire M., Belongie S., Bourdev L., Ross G., James H., Perona P., Ramanan D., Zitnick C. L., Dollár P., "Microsoft COCO: Common Objects in Context", *ArXiv,* ***abs/ 1405.0312***, 2014

[6] Geiger A., Lenz P., Stiller C. and Urtasun R., "Vision meets Robotics: The KITTI Dataset", *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012

[7] Cordts M., Omran M., Ramos S., Rehfeld T., Enzweiler M., Benenson R., Franke U., Roth S., Schiele B., "The Cityscapes Dataset for Semantic Urban Scene Understanding", *ArXiv,* ***abs/ 1604.01685***, 2014

[8] Jain V. and Learned-Miller E., *FDDB: A Benchmark for Face Detection in Unconstrained Settings*, University of Massachusetts, Amherst, 2014, 11 c.

[9] Yang, Shuo and Luo, Ping and Loy, Chen Change and Tang, Xiaoou, "WIDER FACE: A Face Detection Benchmark", *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016

[10] Deng J., and Roussos A., Chrysos G., Ververas E., Kotsia I., Shen J., Zafeiriou S., "The menpo benchmark for multi-pose 2d and 3d facial landmark localisation and tracking", *International Journal of Computer Vision*, **127**:6 (2019), 599–624

[11] Shakhuro V.I., Konushin A.S., "Russian traffic sign images dataset", *Computer Optics*, **40**:2 (2016), 294–300 (In Russian)

[12] Jin Haifeng and Song Qingquan and Hu, Xia, "An Efficient Neural Architecture Search System", *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2019, 1946–1956

[13] Chin-Wei Hsu, Chih-Chung Chang and Chih-Jen Lin, *A practical guide to support vector classification*, National Taiwan University, Taipei, 2010, 16 c.

[14] Aurelien Geron, *Hands-On Mashine Learning with Scikit-Learn, Keras, and TensorFlow, 2nd Edition*, O'Reilly Media, Inc., Sebastopol, California, 2019, 856 c.

[15] James Bergstra, Yoshua Bengio, "Random Search for Hyper-Parameter Optimization", *Machine Learning Research*, **13** (2012), 281–305

[16] Bergstra, J., Yamins, D., Cox, D. D., "Making a Science of Model Search: Hyperparameter Optimization in Hundreds of Dimensions for Vision Architectures.", *TProc. of the 30th International Conference on Machine Learning (ICML 2013)*, 2013

[17] Larsen J., Hansen L.K., Svarer C., Ohlson B.O.M., "Design and regularization of neural networks: the optimal use of a validation set", *IEEE Signal Processing Society Workshop*, 1996, 62–71

[18] Maclaurin D., Duvenaud D., Adams R. P., "Gradient-based Hyperparameter Optimization through Reversible Learning", *ArXiv,* ***abs/ 1502.03492***, 2015

[19] Salimans T., Kingma D.P., Welling M., "Markov chain Monte Carlo and variational inference: Bridging the gap", *ArXiv,* ***abs/ 1410.6460***, 2014

[20] Chen Zhu, Renkun Ni, Zheng Xu, Kezhi Kong, W. Ronny Huang, Tom Goldstein, "GradInit: Learning to Initialize Neural Networks for Stable and Efficient Training", *ArXiv,* ***abs/ 2102.08098***, 2021

[21] MacKay M., Vicol P., Lorraine J., Duvenaud D., Grosse R., "Self-Tuning Networks: Bilevel Optimization of Hyperparameters using Structured Best-Response Functions", *ArXiv,* ***abs/ 1903.03088***, 2019

[22] Bae J., Grosse R., "Delta-STN: Efficient Bilevel Optimization for Neural Networks using Structured Response Jacobians", *ArXiv,* ***abs/ 2010.13514***, 2020

[23] Franceshi L., Frasconi P., Salso S., Grazzi R., Pontil M., "Bilevel Programming for Hyperparameter Optimization and Meta-Learning", *ArXiv,* ***abs/ 1806.04910***, 2018

[24] Correia A. H.C., Worrall D. E., Bondesan R., "Neural Simulated Annealing", *ArXiv,* ***abs/ 2203.02201***, 2022

Регуляризация свёрточной нейронной сети сингулярным разложением для обучения на малых выборках

Н.В. Ваулин[1]

Задача обучения на малой выборке решается за счет оптимизации параметров предобученной сверточной нейронной сети с ограничениями на веса сверток. Регуляризация весов осуществляется путем представления параметров свертки в виде набора сингулярных разложений матриц и обучения только сингулярных значений разложений. Исследуется влияние сингулярных значений на качество модели и динамика их изменений во время обучения. Приведены результаты применения предложенной регуляризации в сравнении с другими методами в задаче классификации рентгенологических снимков.

Ключевые слова: машинное обучение, свёрточные сети, разложение на сингулярные значения.

1. Введение

Автоматизированный анализ медицинских изображений является одной из наиболее перспективной областью применения машинного обучения. Задачу классификации медицинских изображений можно рассматривать в контексте общей проблемы классификации в компьютерном зрении. Здесь так же применимы многие стандартные решения, такие как нейросетевые архитектуры, аугментация данных и стратегии обучения. Однако, отсутствие больших обучающих выборок с одновременной необходимостью высокой обобщающей способности модели, заставляет развивать ряд специфичных решений, апеллирующих к недостатку данных. К ним относится дистилляция, перенос знаний, доменная адаптация – данные методы позволяют обучать качественные модели на малых выборках. Также при решении подобных задач используются нейросетевые архитектуры с малым числом параметров и блоки внимания.

Наиболее распространенный подход к обучению нейронных сетей на малых выборках является использование предобученных моделей. В данном случае возникает вопрос выбора базы для предобучения. Одним из

[1] *Ваулин Николай Владимирович* — аспирант каф. теоретической информатики и дискретной математики, институт математики и информатики МПГУ, e-mail: nvvaulin@gmail.com.

Vaulin Nikolay Vladimirovich — graduate student, MPGU, Faculty of Computer Science and Discrete Mathematics.

вариантов такой базы является ImageNet[14] , что было опробовано в [4]. Однако, отличия в задачах классификации объектов и классификации болезней осложняют процесс адаптации предобученной модели к основной задаче. Таким образом, в [11] было продемонстрированно, что дальнейшее дообучение модели на более близком домене позволяет получить более информативное пространство признаков. Для этих целей авторы статьи собрали выборку из 224,316 КТ снимков грудной клетки с 14 размеченными патологиями на них. Ьодель, предобученную на схожей выборке, можно адаптировать для решения нужной задачи. В [9] обучали последний классификационный слой для адаптации модели к задаче детекции COVID-19. Аналогичную задачу в статье [5] было предложено решить за счет обучения классификатора методом опорных векторов на выходе предпоследнего слоя. В статье [8] также решалась задача классфикации для выявления COVID-19 на снимках грудной клетки. В данном случае, авторы статьи использовали densenet [12], предобученную на chexpert [11], обучили последний классификационный слой и затем всю модель целиком.

Помимо предобучения модели на размеченной выборке, существуют методы, позволяющие задействовать в обучении неразмеченные данные. В [7] обучалась энкодер-декодер модель на снимках человеческого мозга, и затем энкодер использовался для выделения признаков, на которых обучался полносвязный слой в сценарии классификации, используя небольшую выборку. В работе [6] авторы использовали неразмеченную выборку, присвоив разметку на основе названия изображения, что во многом не соответствовало каким-либо патологиям, но хорошо подходило для предобучения модели.

Помимо расширения выборки, эффект переобучения снижают за счет уменьшения количества параметров модели. В статье [3] была предложена архитектура Covid-Net, в которой для снижения количества обучаемых параметров широко использовались PEPX блоки, в которых сначала происходило проецирование признаков на меньшую размерность за счет свертки 1x1, затем размерность увеличивалась сверткой 1x1 и применялась поканальная свертка 3x3, далее размерность снова снижалась и восстанавливалась входная размерность для residual суммирования. Данная архитектура значительно снижала количество весов сверток. В работе [2] было показано, что архитектура из 4-х сверток 32-64-64-128 может превзойти в качестве задаче классификации COVID-19 глубокие архитектуры, такие как Resnet-50 и VGG-19, предобученые на imagenet.

Сингулярное разложение уже широко применяется в нейронных сетях. Например, в задаче распознавания речи удалось снизить количество параметров нейронной сети за счет обнуления малых сингулярных значений [16], уменьшив количество параметров в 8 раз и дообучив, точ-

ность модели практически не снизилась. Аналогично, снижение ранга матрицы использовалось в задаче дообучения (а именно, в self-supervised distillation). В [17] было предложено дистиллировать корреляцию между входными и выходными значениями блока при этом, внутренняя размерность входных и выходных признаков снижалась за счет обнуления сингулярных значений. Дообучение в режиме дистилляции к изначальной модели привело к увеличению точности.

В данной работе предлагается использовать сингулярное разложение весов в задаче переноса знаний при адаптации модели к новой задаче. За счет обучения только сингулярных значений сверточных матриц удается добиться значительного уменьшения количества обучаемых параметров, что позволяет адаптировать модель для конкретной задачи, снизив эффект переобучения.

2. Перенос знаний

Рассмотрим задачу многоклассовой классификации в общем случае. Задача обучения сводится к максимизации правдоподобия параметров модели. Имея достаточно большой размер выборки, задачу можно решить минимизируя эмпирическое математическое ожидание логарифма правдоподобия, что реализуется алгоритмами стохастического градиентного спуска. Однако, в случае, когда количество обучаемых параметров сильно превышает размер обучающей выборки, оценка математического ожидания эмпирическим средним может давать смещенный результат, что проявляется в виде переобучения модели, оптимизация на маленькой выборке в конечном итоге может привести к росту логарифма правдоподобия, что обычно видно на валидационной базе. В данном сценарии широко применяются дополнительные ограничения на обучаемые параметры. Классическим предположением является близость оптимальных весов для схожих задач. Например, имея похожую выборку с большим количеством примеров, модель сначала обучается на ней, затем, осуществляется тонкая настройка параметров, за счет обучения на целевой задаче, используя малый шаг градиентного спуска. В данном случае, веса модели оказываются близки в евклидовой метрике к предобученым весам, что снижает эффект переобучения.

2.1. Сохранение базиса сингулярных векторов

Понятие близости весов можно сформулировать иначе. Рассмотрим веса свертки, их можно представить в виде матрицы (набора матриц). Каждую матрицу сверток исходной модели можно представить в виде сингулярного разложения $M = U\Sigma V^T$. При этом, можно сделать допу-

щение, что при дообучении меняются лишь сингулярные значения, при том что сингулярные вектора сверток остаются неизменными. Данный эффект выражен и без каких-либо дополнительных изменений во время обучения. Чтобы это продемонстрировать, применим матрицы U, V разложения исходных весов к дообученным $\Sigma_{tune} = U^T M_{tune} V$ и к весам модели, обученной со случайной инициализации $\Sigma_{rand} = U^T M_{rand} V$. Затем сравним соотношение суммы квадратов диагональных элементов к сумме квадратов недиагональных $R = \sum_i \Sigma_{ii}^2 / \sum_{i \neq j} \Sigma_{ij}^2$. На рис. 1 изображено распределение этого соотношения. Среднее значение для модели, обученной со случайной инициализации $\overline{R_{rand}} = 0.008$, в то время как для дообученной модели аналогичный параметр более чем в 100 раз выше $\overline{R_{tune}} = 0.91$. Данный эксперимент показывает состоятельность ограничения на дообучаемые параметры в виде фиксации матриц U и V сингулярного разложения сверток.

При вышеописанных ограничениях на веса, близость параметров выражается в схожести сингулярных значений. Забегая вперед, можно отметить, что данное предположение хорошо выполняется: корреляция между исходными сингулярными значениями и значениями дообученной модели при фиксированных U и V составляет 0.9.

2.2. SVD разложение свертки

Веса свертки нейронной сети представляют собой четырехмерный тензор $W \in R^{in,out,h,w}$. Один из способов разложения предполагает объединение размерности, соответствующей выходным каналам, и пространственных размерностей $M \in R^{in,out \cdot h \cdot w}$. Применив разложение на сингулярные значения, матрицу можно представить в виде $W = U \Sigma V^T$, где $U \in R^{out,out}$ и $V \in R^{out \cdot h \cdot w, out \cdot h \cdot w}$ - унитарные матрицы, а Σ - диагональная, с неотрицательными элементами. Зафиксировав матрицы U и V, можно уменьшить количество обучаемых параметров с $in \cdot out \cdot h \cdot w$ до $min(in, out \cdot h \cdot w)$. Данный тип разложения будем называть полным.

Другой способ предполагает представление W в виде набора матриц $W_{ij} \in R^{in,out}$. В данном случае, сингулярное разложение применяется к каждой матрице в отдельности, при этом количество обучаемых параметров будет $min(in, out) \cdot h \cdot w$, данный тип разложения будем называть пространственным, т.к. каждой пространственной компоненте свертки соответствует свое разложение.

Для еще большего снижения количества параметров, можно занулить малые сингулярные значения.

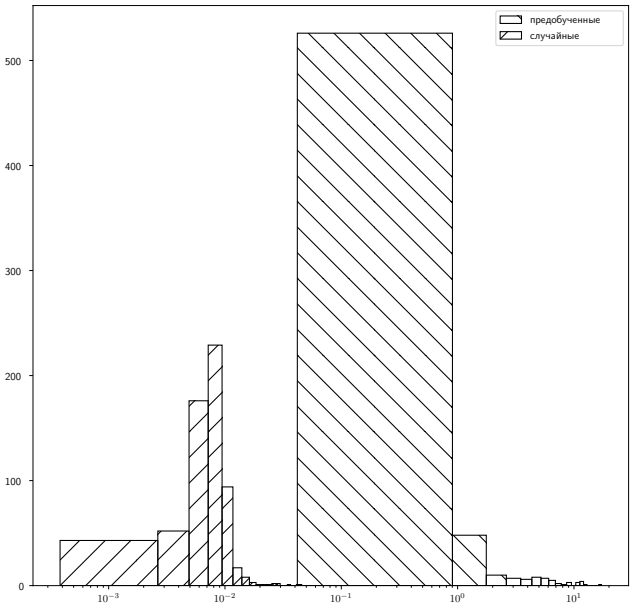


Рис. 1. Гистограмма отношения суммы квадратов диагональных элементов матрицы Σ к недиагональным в разложении по матрицам поворота исходных параметров для дообученной модели и обученной со случайной инициализации.

3. Постановка эксперимента

3.1. Данные

В качестве исходного датасета для предобучения модели была выбрана база NIH Chest X-ray[10]. Датасет содержит 112120 фронтальных рентгеновских изображений 30805 уникальных пациентов с размеченными на них 14 патологиями (каждое изображение может иметь несколько меток), полученных из радиологических отчетов с использованием автоматизированной обработки языка. Четырнадцать патологий грудной клетки включают ателектаз, уплотнение, инфильтрацию, пневмоторакс, отек, эмфизему, фиброз, выпот, пневмонию, уплотнение плевры, кардиомегалию, узелок, объем и грыжу. Стоит отметить, что оригинальные радиологические отчеты не доступны для исследователя, лишь метки, точность которых, как утверждают авторы, составляет не менее 90%.

В качестве целевой задачи была выбрана задача многоклассовой классификации фронтальных рентгенологических снимков грудной слетки на 3 класса: COVID-19 (SARS-CoV-2), пневмония и без патологий. Для данных целей была выбрана обучающая выборка CovidAID[1], состоящая из 478 изображений с COVID-19 взятых из базы [18], 1583 снимков здоровых людей и 4273 снимков людей с пневмонией взятых из [19]. Для всех снимков есть уникальный идентификатор пациента, а точность разметки данных значительно выше, чем в NIH Chest X-ray[10], т.к. использовалась ручная разметка.

Все снимки имеют высокое разрешение, однако, входное разрешение обучаемой модели - квадрат 224 на 224 пиксела, соответственно, разрешение изображений снижалось до входного разрешения модели. В качестве аугментации использовались случайное отражение относительно вертикальной оси и случайная рамка с изменением масштаба в диапазоне от 0.8 до 1.2.

Вся база была разбита на 3 части: обучающая выборка более 88%, валидационная менее 2% и тестовая 10% (на Рис. 2 приведены подробная статистика базы и примеры изображений). При этом, все снимки одного пациента полностью включены в одну из трех подвыборок.

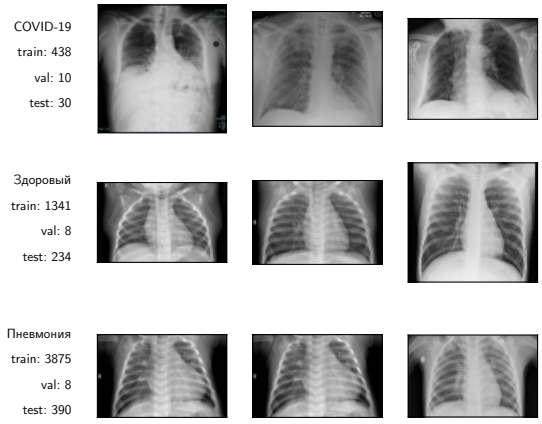


Рис. 2. Примеры изображений и статистика базы CovidAID.

3.2. Модель обучения

В качестве классификатора была выбрана архитектура нейронной сети densenet[12] с размером входного изображения 224 пикселей. В статье [13] была продемонстрирована высокая эффективность модели на базе NIH Chest X-ray[10]. Сеть изначально была обучена на imageNet[14], затем был заменен последний классификационный слой и сеть была дообучена на NIH Chest X-ray с помощью Adam[15] оптимизатора с мультипликатором градиентов 0.001. Полученные веса использовались во всех экспериментах в качестве инициализации.

Так как количество классов в целевой задаче отличалось от исходной, модель обучалась в 2 этапа: на первом этапе, модель, за исключением полносвязного слоя, инициализировалась предобученными весами и обучался только последний полносвязный слой одну эпоху, затем обучалась вся модель в течении 14 эпох. Во всех экспериментах был использован оптимизатор Adam[15] с $\beta_1 = 0.9, \beta_2 = 0.99$, размером батча 16 и мультипликатором градиентов 10^{-3}, причем мультипликатор снижался, если значения функции потерь на валидации не изменялось в течении четырех эпох. Лучшая эпоха выбиралась на основе значения функции потерь на валидационнной выборке.

Так как сингулярные значения матрицы неотрицательные, во время обучения они были представлены в виде квадрата обучаемых параметров (соответственно, инициализировались эти параметры квадратными корнями сингулярных значений).

Для сравнения эффективности предложенного метода, было проведено 5 серий экспериментов.

- Обучение всех параметров весов, как было предложено авторами статьи [1]. Стоит отметить, что параметры обучения не соответствуют референсным, однако использовалась та же база, а результаты на тесте оказались выше, чем было продемонстрировано в статье (roc auc 0.9897 против 0.986), что позволяет использовать данный подход в качестве базового метода для сравнительного анализа (обозначено как **full**).

- Обучение только последнего классификационного слоя по аналогии со статьей [5] (обозначено как **fc-train**).

- Обучение сингулярных значений при полном разложении (обозначено как **full-svd**).

- 2 серии обучения сингулярных значений при пространственном разложении и занулении 0% и 90% сингулярных значений (обозначено как **sp-svd** и **sp-svd9**).

3.3. Тест и мера качества

Для сравнения результатов с базовыми методами была использована подвыборка CovidAID [1]. Мерой качества было выбрано значение площади под ROC-кривой для каждого класса (ROC AUC). Также измерялось отношение верноположительных детекций к общему количеству положительных примеров при пороге, соответствующему доле ложноположительных ответов равной 10^{-3} (tpr @ fmr=10^{-3}). Использование данной меры оправдано для измерения качества детектирования COVID-19, так как в тесте соответствующих примеров значительно меньше, чем примеров с пневмонией или здоровых людей. В таблице 1 приведены результаты для каждой серии экспериментов. Видно, что все 3 метода регуляризация svd разложением в среднем превосходят базовые методы, лишь в некоторых случаях уступая в детектировании COVID-19.

Таблица 1. Сравнение качества на тесте CovidAID

roc auc					
	fc-train	full	full-svd	sp-svd	sp-svd-9
Здоровый	0.9790	0.9837	0.9882	**0.9928**	0.9901
COVID-19	0.9952	**1.0000**	0.9996	0.9999	**1.0000**
Пневмония	0.9663	0.9855	0.9899	**0.9933**	0.9906
Среднее	0.9802	0.9897	0.9926	**0.9953**	0.9936
tpr @ fmr=10−3					
	fc-train	full	full-svd	sp-svd	sp-svd-9
Здоровый	0.2265	0.5641	0.4829	**0.6667**	0.4530
COVID-19	0.5667	**1.0000**	0.9000	0.9333	**1.0000**
Пневмония	0.1667	0.3974	0.3641	**0.5487**	0.4462
Среднее	0.3199	0.6538	0.5823	**0.7162**	0.6330

Стоит отметить, что в данном тесте всего 30 примеров с COVID-19 и меры качества достигали своих максимальных значений в некоторых экспериментах, что свидетельствует о недостаточном размере тестовой подвыборки. Также малый размер валидационной базы мог привести к неоптимальному выбору лучшей эпохи. В целях расширения тестовой выборки разбиение базы на обучающую, тестовую и валидационную было изменено: к исходному тесту и валидации были добавлены примеры за счет сокращения базы обучения, таким образом база была разделена в соотношении 50%, 40%, 10% (обучающая, тестовая и валидационная выборки соответственно), подробная статистика разбиения приведена в табл. 2. Все эксперименты были проведены повторно на новой обучаю-

щей выборке, результаты приведены в табл. 3. Данные результаты во многом повторяют предыдущие, однако, заметны отличия в качестве детектирования COVID-19.

Таблица 2. Статистика обучающей выборки

CovidAID				
	COVID-19	Здоровый	Пневмония	Сумма
Обучение	438	1341	3875	5654
Валидация	10	8	8	26
Тест	30	234	390	654
Сумма	478	1583	4273	6334
Измененный CovidAID				
	COVID-19	Здоровый	Пневмония	Сумма
Обучение	244	757	2168	3169
Валидация	49	152	431	632
Тест	185	674	1674	2533
Сумма	478	1583	4273	6334

Таблица 3. Сравнение качества на расширенном тесте CovidAID

roc auc					
	fc-train	full	full-svd	sp-svd	sp-svd-9
Здоровый	0.9841	0.9908	**0.9943**	**0.9943**	0.9928
COVID-19	0.9903	0.9998	**0.9999**	0.9998	0.9998
Пневмония	0.9696	0.9908	0.9945	**0.9949**	0.9938
Среднее	0.9813	0.9938	0.9962	**0.9964**	0.9955
tpr @ fmr=$10^{-}3$					
	fc-train	full	full-svd	sp-svd	sp-svd-9
Здоровый	0.4525	0.5786	0.6736	**0.6751**	0.5445
COVID-19	0.3514	0.9622	**0.9892**	0.9405	0.9838
Пневмония	0.1147	0.2987	0.3363	**0.5299**	0.4217
Среднее	0.3062	0.6132	0.6664	**0.7152**	0.6500

3.4. Связь качества обучения и размера выборки

Для определения зависимости значения мер качества от размера обучающей выборки, были проведены эксперименты на базе, уменьшенной

в 2,4 и 8 раз. Для всех серий были использованы одинаковые подвыборки обучающей базы. В соответствии с результатами, приведенными в табл. 4, предложенная регуляризация показывает значительные преимущества при уменьшении базы. Однако, по графику изображенному на рис. 3 видно, что при уменьшении базы полное разложение показывает лучшие результаты, чем пространственное. В то же время, разрыв между обучением всех сингулярных значений и отброс 90% значений, снижается при уменьшении базы. Таким образом заметна зависимость качества обучения от количества обучаемых параметров, полное разложение имеет сильно меньше обучаемых параметров, и показывает себя лучше при уменьшении базы, в то время как качество при обучении всех параметров резко падает.

Таблица 4. Сравнение качества модели при уменьшении обучающей базы на расширенном тесте CovidAID

roc auc					
Размер подвыборки	fc-train	full	full-svd	sp-svd	sp-svd-9
1	0.9813	0.9938	0.9962	**0.9964**	0.9955
1/2	0.9771	0.9889	0.9952	**0.9958**	0.9937
1/4	0.9762	0.9904	0.9937	**0.9941**	0.9928
1/8	0.9701	0.9645	**0.9916**	0.9863	0.9907
tpr @ fmr=$10^{-}3$					
Размер подвыборки	fc-train	full	full-svd	sp-svd	sp-svd-9
1	0.3062	0.6132	0.6664	**0.7152**	0.6500
1/2	0.2244	0.5238	0.6635	**0.6740**	0.5716
1/4	0.1958	0.4336	0.4928	**0.5674**	0.5373
1/8	0.2395	0.2544	**0.4730**	0.3942	0.4721

4. Анализ результатов

Предложенный подход позволяет регуляризовать параметры сети и избежать эффекта переобучения даже на небольшой обучающей выборке, в то же время, оптимальный метод разложения весов зависит от размера выборки. Метод позволяет гибко уменьшать количество обучаемых параметров, если это требуется при имеющихся данных для обучения. Также метод позволяет уменьшить общее количество параметров сети: при отбрасывании 90% параметров сети, качество модели снижалось незначительно, таким образом можно изменить архитектуру сети в целях снижения вычислительной сложности или для сжатия параметров

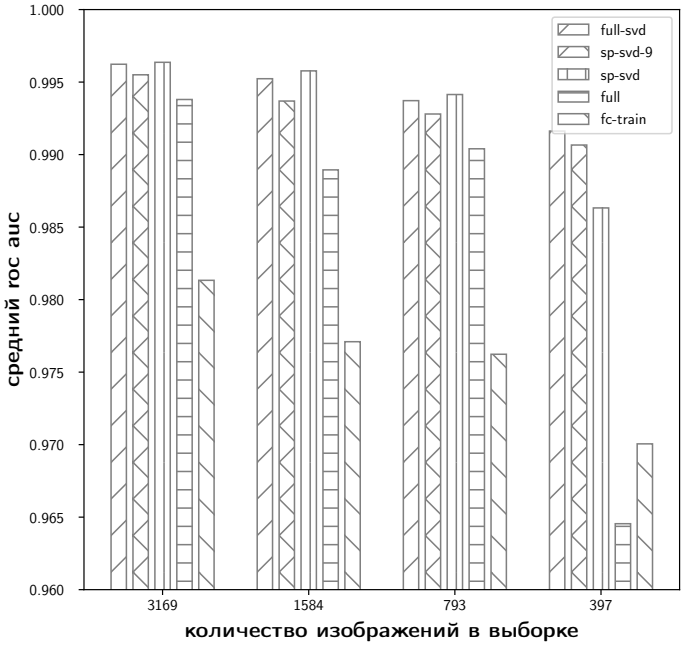


Рис. 3. Зависимость качетсва обучения (roc auc на расширенном CovidAID) при уменьшении базы в 1, 2, 4 и 8 раз.

сети. Данный результат во многом повторяет [16], но уже в применении к сверточным сетям.

4.1. Динамика сингулярных значений

Рассмотрим взаимосвязь между исходными и дообученными сингулярными значениями. Так как после светки всегда используется батч нормализация, стоит нормализовать значения, в данном случае, все сингулярные значения свертки делились на максимальное значение. На рис. 4 приведены гистограммы сингулярных значений и их изменений. Видно, что малые значения еще больше уменьшились, что свидетельствует о вырождении шумовых компонент, при этом усиливались лишь значимые компоненты, нормализованные значения которых больше 0.4. Любопытным является тот факт, что более половины (55%) сингулярных значений занулились, при том, что изначально околонулевых значений было менее 1%.

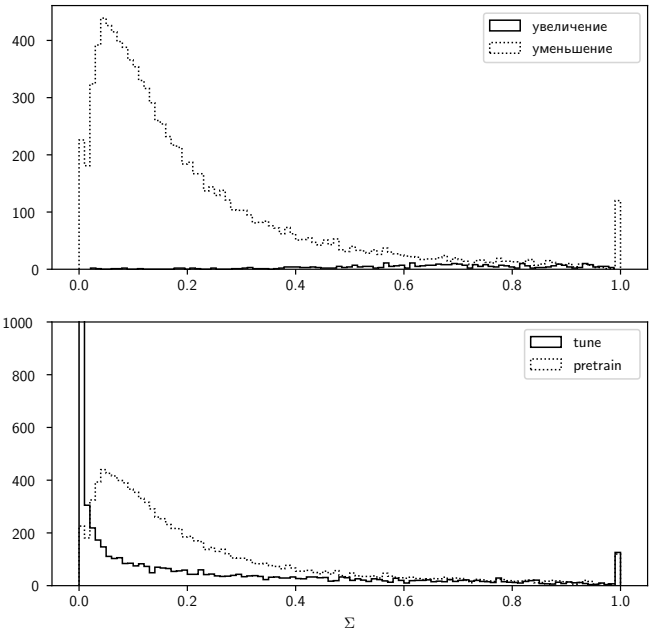


Рис. 4. Динамика изменений (сверху) и распределение (снизу) сингулярных значений предобученной модели (pretrain) и модели, дообученной на целевой задаче (tune).

4.2. Дисперсия обучения

Дополнительным преимуществом метода является его стабильность во время обучения. На рис. 5 видно, что метод полного разложения дает значительно более стабильные результаты в сравнении с обучением всех параметров: дисперсия значений функции потерь на разных эпохах более чем в 3 раза ниже (0.015 против 0.047), однако наиболее стабилен остается метод обучения последнего полносвязного слоя (дисперсия 0.009). Меньший разброс значений означает меньшую зависимость качества модели от выбора валидационной базы, что актуально при малом количестве данных, ввиду сложности сбора репрезентативной валидационнной выборки.

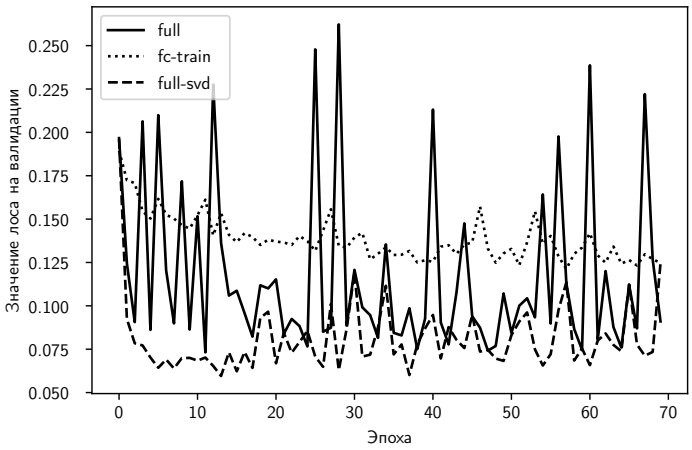


Рис. 5. Значения функции потерь на тестовой выборке в зависимости от эпохи.

5. Заключение

В работе предложен новый подход регуляризации параметров. Параметризация весов свертки в виде матриц сингулярного разложения и последующее дообучение лишь сингулярных значений ведет к снижению эффекта переобучения. Использование данного метода привело к повышению точности результатов на целевой задаче даже при кратном уменьшении размера обучающей выборки. Метод показал высокую стабильность во время обучения, что актуально в условиях отсутствия репрезентативной валидационной базы. Возможность уменьшения количества обучаемых параметров за счет зануления малых сингулярных значений позволяет эффективно использовать данный метод в широком диапазоне размеров обучающих выборок.

Побочным, но, несомненно, полезным, результатом метода является снижение ранга матриц свертки, что можно использовать для уменьшения количества параметров сети или для ее ускорения. Была продемонстрирована возможность эффективного сжатия сверточной сети счет обнуления сингулярных значений.

Список литературы

[1] Mangal, Arpan and Kalia, Surya and Rajgopal, Harish and Rangarajan, Krithika and Namboodiri, Vinay and Banerjee, Subhashis and Arora,

Chetan, "CovidAID: COVID-19 detection using chest X-ray", *arXiv preprint arXiv:2004.09803*, 2020.

[2] Haque, Khandaker Foysal and Abdelgawad, Ahmed, "A deep learning approach to detect COVID-19 patients from chest X-ray images", *AI*, **1**:3 (2020), 418–435.

[3] Wang, Linda and Lin, Zhong Qiu and Wong, Alexander, "Covid-net: A tailored deep convolutional neural network design for detection of covid-19 cases from chest x-ray images", *Scientific Reports*, **10**:1 (2020), 1–12.

[4] Raghu, Maithra and Zhang, Chiyuan and Kleinberg, Jon and Bengio, Samy, "Transfusion: Understanding transfer learning for medical imaging", *arXiv preprint arXiv:1902.07208*, 2019.

[5] Sethy, Prabira Kumar and Behera, Santi Kumari, "Detection of coronavirus disease (covid-19) based on deep features", 2020.

[6] Alzubaidi, Laith and Al-Amidie, Muthana and Al-Asadi, Ahmed and Humaidi, Amjad J and Al-Shamma, Omran and Fadhel, Mohammed A and Zhang, Jinglan and Santamarí a, J and Duan, Ye, "Novel Transfer Learning Approach for Medical Imaging with Limited Labeled Data", *Cancers*, **13**:7 (2021), 1590.

[7] Chen, Min and Shi, Xiaobo and Zhang, Yin and Wu, Di and Guizani, Mohsen, "Deep features learning for medical image analysis with convolutional autoencoder neural network", *IEEE Transactions on Big Data*, 2017.

[8] Singh, Shrinjal and Sapra, Piyush and Garg, Aman and Vishwakarma, Dinesh Kumar, *2021 5th International Conference on Computing Methodologies and Communication (ICCMC)*, 2021.

[9] Minaee, Shervin and Kafieh, Rahele and Sonka, Milan and Yazdani, Shakib and Soufi, Ghazaleh Jamalipour, "Deep-covid: Predicting covid-19 from chest x-ray images using deep transfer learning", *Medical image analysis*, **65** (2020), 101794.

[10] Wang, X and Peng, Y and Lu, L and Lu, Z and Bagheri, M and Summers, R, *IEEE CVPR*, 2017.

[11] Irvin, Jeremy and Rajpurkar, Pranav and Ko, Michael and Yu, Yifan and Ciurea-Ilcus, Silviana and Chute, Chris and Marklund, Henrik and Haghgoo, Behzad and Ball, Robyn and Shpanskaya, Katie and others, *Proceedings of the AAAI conference on artificial intelligence*, **33**, 2019.

[12] Gao Huang and Zhuang Liu and Laurens van der Maaten and Kilian Q. Weinberger, "Densely Connected Convolutional Networks", 2018.

[13] Rajpurkar, Pranav and Irvin, Jeremy and Zhu, Kaylie and Yang, Brandon and Mehta, Hershel and Duan, Tony and Ding, Daisy and Bagul, Aarti and Langlotz, Curtis and Shpanskaya, Katie and others, "Chexnet: Radiologist-level pneumonia detection on chest x-rays with deep learning", *arXiv preprint arXiv:1711.05225*, 2017.

[14] Deng, Jia and Dong, Wei and Socher, Richard and Li, Li-Jia and Li, Kai and Fei-Fei, Li, *2009 IEEE conference on computer vision and pattern recognition*, 2009.

[15] Kingma, Diederik P and Ba, Jimmy, "Adam: A method for stochastic optimization", *arXiv preprint arXiv:1412.6980*, 2014.

[16] Xue, Jian and Li, Jinyu and Gong, Yifan, *Interspeech*, 2013.

[17] Lee, Seung Hyun and Kim, Dae Ha and Song, Byung Cheol, *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018.

[18] Joseph Paul Cohen and Paul Morrison and Lan Dao and Karsten Roth and Tim Q Duong and Marzyeh Ghassemi, "COVID-19 Image Data Collection: Prospective Predictions Are the Future", *arXiv 2006.11988*, 2020.

[19] Kermany, Daniel S and Goldbaum, Michael and Cai, Wenjia and Valentim, Carolina CS and Liang, Huiying and Baxter, Sally L and McKeown, Alex and Yang, Ge and Wu, Xiaokang and Yan, Fangbing and others, "Identifying medical diagnoses and treatable diseases by image-based deep learning", *Cell*, **172**:5 (2018), 1122–1131.

Neural network regularization via SVD for small dataset tasks

Vaulin N.V.

Consider convolutional neural network optimization problem using small dataset. The proposed method fine-tune pretrained neural network with specigic constrains for convolutional kernels. Pretrained weights are decomposed using SVD. During fine-tuning the only singular values are trained. In the paper are investigated the dinamic of singular values during training process and its influence on quality of resulting model. The method are comapared with other approaches in rengenological images classsification problm.

Keywords: deep learning, convolutional neural networks, singular value decomposition.

References

[1] Mangal, Arpan and Kalia, Surya and Rajgopal, Harish and Rangarajan, Krithika and Namboodiri, Vinay and Banerjee, Subhashis and Arora, Chetan, "CovidAID: COVID-19 detection using chest X-ray", *arXiv preprint arXiv:2004.09803*, 2020.

[2] Haque, Khandaker Foysal and Abdelgawad, Ahmed, "A deep learning approach to detect COVID-19 patients from chest X-ray images", *AI*, **1**:3 (2020), 418–435.

[3] Wang, Linda and Lin, Zhong Qiu and Wong, Alexander, "Covid-net: A tailored deep convolutional neural network design for detection of covid-19 cases from chest x-ray images", *Scientific Reports*, **10**:1 (2020), 1–12.

[4] Raghu, Maithra and Zhang, Chiyuan and Kleinberg, Jon and Bengio, Samy, "Transfusion: Understanding transfer learning for medical imaging", *arXiv preprint arXiv:1902.07208*, 2019.

[5] Sethy, Prabira Kumar and Behera, Santi Kumari, "Detection of coronavirus disease (covid-19) based on deep features", 2020.

[6] Alzubaidi, Laith and Al-Amidie, Muthana and Al-Asadi, Ahmed and Humaidi, Amjad J and Al-Shamma, Omran and Fadhel, Mohammed A and Zhang, Jinglan and Santamarí a, J and Duan, Ye, "Novel Transfer Learning Approach for Medical Imaging with Limited Labeled Data", *Cancers*, **13**:7 (2021), 1590.

[7] Chen, Min and Shi, Xiaobo and Zhang, Yin and Wu, Di and Guizani, Mohsen, "Deep features learning for medical image analysis with convolutional autoencoder neural network", *IEEE Transactions on Big Data*, 2017.

[8] Singh, Shrinjal and Sapra, Piyush and Garg, Aman and Vishwakarma, Dinesh Kumar, *2021 5th International Conference on Computing Methodologies and Communication (ICCMC)*, 2021.

[9] Minaee, Shervin and Kafieh, Rahele and Sonka, Milan and Yazdani, Shakib and Soufi, Ghazaleh Jamalipour, "Deep-covid: Predicting covid-19 from chest x-ray images using deep transfer learning", *Medical image analysis*, **65** (2020), 101794.

[10] Wang, X and Peng, Y and Lu, L and Lu, Z and Bagheri, M and Summers, R, *IEEE CVPR*, 2017.

[11] Irvin, Jeremy and Rajpurkar, Pranav and Ko, Michael and Yu, Yifan and Ciurea-Ilcus, Silviana and Chute, Chris and Marklund, Henrik and Haghgoo, Behzad and Ball, Robyn and Shpanskaya, Katie and others, *Proceedings of the AAAI conference on artificial intelligence*, **33**, 2019.

[12] Gao Huang and Zhuang Liu and Laurens van der Maaten and Kilian Q. Weinberger, "Densely Connected Convolutional Networks", 2018.

[13] Rajpurkar, Pranav and Irvin, Jeremy and Zhu, Kaylie and Yang, Brandon and Mehta, Hershel and Duan, Tony and Ding, Daisy and Bagul, Aarti and Langlotz, Curtis and Shpanskaya, Katie and others, "Chexnet: Radiologist-level pneumonia detection on chest x-rays with deep learning", *arXiv preprint arXiv:1711.05225*, 2017.

[14] Deng, Jia and Dong, Wei and Socher, Richard and Li, Li-Jia and Li, Kai and Fei-Fei, Li, *2009 IEEE conference on computer vision and pattern recognition*, 2009.

[15] Kingma, Diederik P and Ba, Jimmy, "Adam: A method for stochastic optimization", *arXiv preprint arXiv:1412.6980*, 2014.

[16] Xue, Jian and Li, Jinyu and Gong, Yifan, *Interspeech*, 2013.

[17] Lee, Seung Hyun and Kim, Dae Ha and Song, Byung Cheol, *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018.

[18] Joseph Paul Cohen and Paul Morrison and Lan Dao and Karsten Roth and Tim Q Duong and Marzyeh Ghassemi, "COVID-19 Image Data Collection: Prospective Predictions Are the Future", *arXiv 2006.11988*, 2020.

[19] Kermany, Daniel S and Goldbaum, Michael and Cai, Wenjia and Valentim, Carolina CS and Liang, Huiying and Baxter, Sally L and McKeown, Alex and Yang, Ge and Wu, Xiaokang and Yan, Fangbing and others, "Identifying medical diagnoses and treatable diseases by image-based deep learning", *Cell*, **172**:5 (2018), 1122–1131.

Градиентная маска и обобщения нейронной сети

Л. Цзян [1]

В рамках практического применения нейронных сетей количество параметров в сети намного больше, чем количество выборок в наборе данных, однако сеть по-прежнему имеет хорошие характеристики обобщения. Традиционно считается, что такие сверхпараметризованные и невыпуклые модели могут легко попадать в локальные минимумы при поиске оптимального решения и показывать плохую производительность обобщения, но на самом деле это не так. Хотя при некоторых условиях регуляризации возможно эффективно контролировать ошибку обобщения сети, по-прежнему трудно объяснить проблему обобщения для больших сетей. В данной статье мы определяем разницу между этапом переобучения и этапом изучения признаков путем количественной оценки влияния обновления одной выборки во время градиентного спуска на весь процесс обучения, выявив, что нейронные сети обычно меньше влияют на другие образцы на этапе переобучения. Кроме того, мы используем информационную матрицу Фишера для маскировки градиента, полученного в процессе обратного распространения, тем самым замедляя поведение нейронной сети при переобучении и улучшая производительность обобщения нейронной сети.

Ключевые слова: Нейронные сети, обобщение, переобучение, информация фишера.

1. Введение: Обобщающая способность нейронных сетей

Обобщение относится к способности модели правильно предсказывать данные, которые она никогда раньше не видела. Изучение способности сверх-параметризованных нейронных сетей к обобщению уже давно представляет интерес в рамках сферы машинного обучения, поскольку идет вразрез с положениями классической теории обучения.

С одной стороны, сверх-параметризованные нейронные сети способны сходиться к нулевым потерям на подавляющем большинстве обучающих наборов данных. Например, работа [1] доказывает, что нейронные

[1] *Цзян Лэй* — аспирант каф. математической теории интеллектуальных систем мех.-мат. ф-та МГУ, e-mail: kiwee@outlook.com

Jiang Lei — graduate student, Lomonosov Moscow State University, Faculty of Mechanics and Mathematics, Chair of Mathematical Theory of Intellectual Systems.

сети способны сходиться даже на случайных данных (зашумленные данные, случайные метки). Это указывает на то, что нейронная сеть достаточно приспособлена к обучающим данным. С другой стороны, нейронные сети, которые особенно подходят для тестового набора данных, могут по-прежнему обеспечивать хорошую производительность теста.

С точки зрения теории, некоторые исследования начинаются с мощности модели нейронных сетей: ошибка обобщения оценивается путем изучения взаимосвязи между мощностью модели и обучающими данными. Типичные методы: размерность Вапника — Червоненкиса (VC dimension) [2, 3], Радемахеровская сложность (Rademacher complexity) [4] и т.д. Другие придерживаются байесовской точки зрения, изучая и измеряя разницу между априорным и апостериорным распределениями модели, чтобы понять ошибку обобщения. Типичная теория — теория Байеса для ВПК-обучения (PAC-bayes) [5, 6, 7]. Напротив, теория PAC-bayes носит общий характер и поэтому применима к различным архитектурам нейронных сетей и наборам данных. Но есть и недостатки, а именно — зависимость от выбора априорного значения [8], в то время как необоснованный априорный выбор приводит к неверным границам обобщения.

На практике использовались различные технические методы для смягчения переобучающего поведения нейронных сетей, тем самым улучшая их способность к обобщению. Типичные технические методы включают:

1) Досрочное завершение: использование проверочного набора для отслеживания переобучения во время обучения и прерывание обучение, когда нейронная сеть входит в стадию доминирования переобучения.

2) Увеличение данных [9]: перед тем, как данные поступают в нейронную сеть, их всячески улучшают с целью увеличить сложность запоминания образцов нейронной сетью.

3) Использование случайности: например, Dropout [10] случайным образом отбрасывает нейроны во время прямого распространения сети.

4) Регуляризация веса [11].

5) Плоскостность: исследования [12, 13, 14] доказали, что гладкость вокруг локального минимума, к которому в итоге сходится нейронная сеть, влияет на производительность обобщения самой сети, а более плоский локальный минимум может улучшить производительность обобщения. Следовательно, явное вынуждение нейронной сети к поиску плоского локального минимума [15] в процессе оптимизации также является техническим методом, который может улучшить производительность обобщения.

2. Вклад

Наш вклад:

1. Мы измеряем влияние обновления одной выборки на другие в процессе градиентного спуска с помощью расстояния Кульбака — Лейблера (РКЛ; KL) и обнаруживаем, что влияние нейронной сети на процесс обучения на этапе выборки памяти (переобучения) в целом меньше, чем на этапе обучения признакам.

2. Мы обнаруживаем, что размер диагональных элементов информационной матрицы Фишера тесно связан с переобучением: во время обратного распространения мы используем информационную матрицу Фишера, чтобы замаскировать градиент веса, замедляя переобучение нейронной сети, тем самым улучшая эффективность обобщения сети.

3. Методология

В качестве примера возьмем задачу множественной классификации: задан набор данных $D = \{(x_1, y_1), \ldots, (x_n, y_n)\}$, где x_i — данные, y_i — метка one-hot. Пусть $f(x; \theta)$ — это нейронная сеть. Мы рассматриваем нейронную сеть как вероятностную модель, а цель оптимизации состоит в том, чтобы максимизировать функцию правдоподобия:

$$\arg\max_{\theta} = \prod_{i=1}^{n} \prod_{j=1}^{K} f^j(x_i; \theta)^{y_i^j} \tag{1}$$

где K — количество категорий в задаче мультиклассификации, y_i^j — i-ый индекс класса образца, $f^j(x_i; \theta)$ — вероятность j-го класса на выходе нейронной сети для выборки x_i. Максимизация функции правдоподобия эквивалентна минимизации отрицательного логарифмического правдоподобия:

$$\arg\min_{\theta} = -\sum_{i=1}^{n} \sum_{j=1}^{K} y_i^j \log f^j(x_i; \theta) \tag{2}$$

Пусть функция потерь — это $\mathcal{L}(x, y, \theta)$, тогда $\sum_{j=1}^{K} y^j \log f^j(x; \theta) = \mathcal{L}(x, y, \theta)$ — функция потерь перекрестной энтропии.

Сначала мы исследуем, насколько «знания», полученные нейронной сетью из одной выборки, влияют на другие во время градиентного спуска. Пусть случайная выборка — это $(x_i, y_i) \in D$, тогда знания, полученные нейронной сетью из выборки x_i могут быть выражены как градиент — т.е. $g_i = \frac{\nabla \mathcal{L}((x_i, y_i), \theta)}{\nabla \theta}$. Тогда влияние на другие выборки можно измерить по следующей формуле:

$$KL(f(x; \theta) || f(x; \theta - \epsilon g_i)) \tag{3}$$

где KL - это Расстояние Кульбака — Лейблера.

Так как мы считаем нашу нейронную сеть как вероятностную модель с параметрами θ, вследствие этого, информационная матрица Фишер F_θ в θ определяется следующим образом:

$$F_\theta = \mathbb{E}_f[(\nabla \log f(x;\theta))(\nabla \log f(x;\theta))^T]$$

Теорема 1 ([26]). *При ϵ стремится к нулю, имеет место равенство*

$$KL(f(x;\theta)||f(x;\theta-\epsilon g_i)) = \frac{1}{2}\epsilon^2 (g_i)^T F_\theta g_i + O(\epsilon^3) \tag{4}$$

Доказательство. Для упрощения обозначений, пишем f_θ как $f(x;\theta)$, и $f_{\theta'}$ как $f(x;\theta-\epsilon g_i)$. Рассмотрим ряд Тейлора $KL(f(x;\theta)||f(x;\theta-\epsilon g_i))$ в точке θ:

$$\begin{aligned} KL(f_\theta||f_{\theta'}) = \quad & KL \quad (f_\theta||f_\theta) - \epsilon {g_i}^T \nabla_{\theta'} KL(f_\theta||f_{\theta'})|_{\theta'=\theta} \\ + \quad & \frac{1}{2}\epsilon^2 {g_i}^T [\nabla^2_{\theta'} KL(f_\theta||f_{\theta'})|_{\theta'=\theta}] g_i + O(\epsilon^3) \end{aligned}$$

Ясно, что первый член равен нулю, так как расстояние KL между одинаковыми распределениями равно нулю. Дальше, рассмотрим первую и вторую производную РКЛ в точке $\theta' = \theta$:

$$\nabla_{\theta'} KL(f_\theta||f_{\theta'})|_{\theta'=\theta} = -\mathbb{E}_{f_\theta}[\nabla_{\theta'} \log f_{\theta'}|_{\theta'=\theta}] = -\mathbb{E}_{f_\theta}[\frac{1}{f_\theta}(\nabla_\theta f_\theta|_{\theta'=\theta})] = 0 \tag{5}$$

и

$$\nabla^2_{\theta'} KL(f_\theta||f_{\theta'})|_{\theta'=\theta} = -\mathbb{E}_{f_\theta}[(\nabla^2_{\theta'} \log f_{\theta'}|_{\theta'=\theta})] = \mathbb{E}_{f_\theta}[H_{\log f_\theta}] = F_\theta \tag{6}$$

Из (3.5) и (3.6) получим: $KL(f(x;\theta)||f(x;\theta-\epsilon g_i)) = \frac{1}{2}\epsilon^2 (g_i)^T F_\theta g_i + O(\epsilon^3)$ □

На практике, учитывая набор данных, мы используем эмпирический метод Фишера(Empirical Fisher) для аппроксимации матрицы Фишера [16]:

$$F_\theta = \frac{1}{n}\sum_{i=1}^{n}[(\nabla_\theta \log f(x_i;\theta))(\nabla_\theta \log f(x_i;\theta))^T] \tag{7}$$

Из формулы 4 видно, что влияние информации, полученной нейронной сетью из одной выборки, на все обучение связано с информацией Фишера о текущих параметрах нейронной сети.

В работе [17, 18, 19] были выявлены две фазы обучения нейронной сети: фаза быстрого обучения (rapid learning) и фаза итеративной детализации (iterative refinement). На этапе быстрого обучения сеть сначала

изучает функции «общего назначения» (обучение представлений), которые оказывают существенное влияние на задачу классификации. На этом этапе значение функции потерь быстро уменьшается, а нейронная сеть может быстро и правильно классифицировать большинство простых образцов. По мере обучения нейронная сеть постепенно входит в стадию итеративной детализации: настраивает изученные функции, предоставляя возможность идентифицировать сложные образцы. Работа [20] подчеркивает, что на этапе итеративной детализации также увеличивается риск переобучения нейронной сети. Логично, что нейронная сеть на этапе переобучения запоминает сами образцы данных, а не изучает значимые признаки. Необходимо учитывать, что этап переобучения просто запоминает образцы и не помогает процессу обучения.

Пусть g_o — градиент, соответствующий образцу памяти, а g_f — градиент, соответствующий обучению признаков. По сравнению с обучением по признакам ожидаемое значение формулы 4 в процессе запоминания образцов должно быть меньше ожидаемого значения на этапе обучения по признакам. Имеем:

$$\mathbb{E}_{g_o \sim \text{overfitting}}[\epsilon^2 g_o^T F_\theta g_o] < \mathbb{E}_{g_f \sim \text{feature}}[\epsilon^2 g_f^T F_\theta g_f] \tag{8}$$

Мы проверяем достоверность утверждения, отслеживая формулу 8 во время обучения: эксперименты на наборе данных CIFAR-100 с использованием глубокой остаточной сверточной нейронной сети ResNet-18. Мы задействуем случайные метки, чтобы заставить нейронную сеть запомнить образцы данных. Результаты представлены на рисунке 1.

Во-первых, видим, что в процессе обучения значение формулы 4 постепенно уменьшается с увеличением времени обучения, отражая переход от этапа быстрого обучения к этапу итеративной детализации. Во-вторых, мы обнаружили, что при запоминании сетью случайных меток, влияние на другие образцы меньше, чем при обычном обучении.

В формуле 4 , поскольку масштаб информационной матрицы Фишера равен квадрату числа параметров сети, это делает неприемлемым расчет и хранение матрицы Фишера в современных больших нейронных сетях. Поэтому мы обращаемся к работе [21, 22]. Возьмем матрицу Фишера, используемую в формуле 4 , и используем ее диагональ в качестве аппроксимации, обозначив диагональные элементы F_θ как $a = (a_{11}, \ldots, a_{mm})$. Тогда формулу 4 можно записать следующим образом:

$$\epsilon^2 g_i^T F_\theta g_i = \sum_{j=1}^{m} a_{jj} g_{ij}^2 \tag{9}$$

Мы можем использовать формулу 8 для анализа того, сколько информации об образце памяти содержится в каждом компоненте гради-

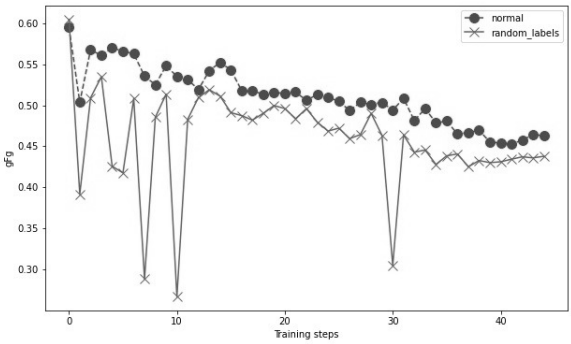


Рис. 1. Значения формулы 4 , соответствующие сетям, обученным с использованием случайных, а также обычных меток

ента g_i в процессе обратного распространения. Формулу 8 можно рассматривать как скалярное произведение вектора $g_i \odot g_i$ (здесь символ $\odot$ - произведение Адамара) и вектора, образованного диагональю матрицы Фишера. Ниже, компонент градиента g_i, соответствующий низкой информации Фишера, с большей вероятностью будет смещен в сторону образца памяти. То есть, мы можем предотвратить переобучение, замаскировав некоторые из этих обновлений параметров.

4. Эксперимент

В данном разделе мы сначала проверяем, может ли механизм маски замедлить явление переобучения, путем экспериментальной количественной оценки взаимосвязи между значением формулы 4 и потерями теста. После этого мы провели эксперименты с различными типами наборов данных и убедились, что механизм маски действительно может улучшить способность сети к обобщению.

4.1. Численный анализ

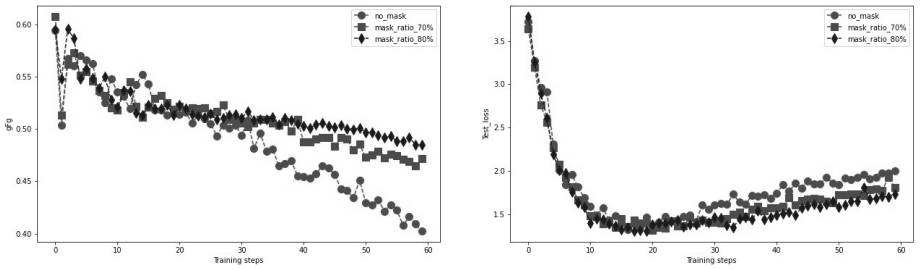
В предыдущем разделе мы обсуждали связь информационной матрицы Фишера с переобучением, а в этом экспериментируем с реальными наборами данных. Мы используем остаточную нейронную сеть ResNet-18 [23], обученную на наборе данных CIFAR-100 [24], который содержит 50 000 обучающих изображений и 10 000 тестовых изображений (всего 100 категорий). Мы маскируем некоторые компоненты в градиенте весов, которые называем механизмом маски градиента, и используем, чтобы наблюдать, может ли механизм маски смягчить явление переобучения.

Сначала сортируем диагональные элементы матрицы Фишера от больших к меньшим, а затем выбираем порог $k \in \{a_{11}, \ldots, a_{mm}\}$, согласно заданному проценту. Формула обновления для градиентного спуска выражается как:

$$\theta^{t+1} = \theta^t - \alpha(g^t \odot M^t), M_i^t = \begin{cases} 1, a_{ii}^t \geq k \\ 0, a_{ii}^t < k \end{cases}, \quad i = 1, \ldots, m \tag{10}$$

где g^t — это градиент весов в момент времени t, M^t — маска, a_{ii}^t — i-ый элемент диагональной матрицы Фишера в точке t.

В качестве порогов выбираем первые 70% и 80% значения диагональной матрицы Фишера, и обучаем сеть градиентным спуском по формуле 10. На рисунке 2 изображена связь между механизмом маски и переобучением. Мы видим, что все три сети в эксперименте перешли в переобучение примерно через 15 эпох. Но после использования механизма маски значение формулы 4 уменьшается медленнее (рисунка 2(a)), значительно замедляя скорость переобучения (рисунка 2(b)). Это доказывает, что механизм маски может эффективно подавлять память нейронной сети об образцах, заставляя нейронную сеть уделять больше внимания изучению признаков.



(a) (b)

Рис. 2. Слева (а) показаны значения формулы 4 , соответствующие трем сетям в процессе обучения; справа (б) показаны тестовые потери в каждый тренировочный момент

4.2. Механизм маски, улучшающий производительность обобщения

Мы проверяем способность механизма маски к обобщению на нескольких различных сетевых архитектурах, а также на разных наборах данных классификации. Для небольших задач (CIFAR-100) мы используем

остаточную сеть ResNet-18 с примерно 12 миллионами параметров, а для больших задач — ImageNet [25], задействовав остаточную нейронную сеть ResNet-50 [23] с 25 миллионами параметров. Из-за их различной сложности для небольших задач мы используем коэффициент маски 0,75, а для больших — 0,25. В целях экономии вычислительных ресурсов мы определяем диагональную матрицу Фишера после каждой эпохи (epoch) в процессе обучения и используем как основу для выбора маски для следующей эпохи обучения. Результаты, представленные в таблице 1, демонстрируют, что способность сети к обобщению улучшилась после использования механизма маски.

Модель	CIFAR-100	Imagenet
Обычная ResNet-18/50	78.2	75.5
ResNet-18/50 с маской	**80.26**	**76.01**

Таблица 1. Лучшая точность (%) ResNet-18 с/без маски на CIFAR-100 и ResNet-50 с/без маски на Imagenet

5. Заключение

В данной статье мы определяем разницу между этапом переобучения и этапом изучения признаков путем количественной оценки влияния обновления одной выборки во время градиентного спуска на весь процесс обучения, выявив, что нейронные сети обычно меньше влияют на другие образцы на этапе переобучения. Кроме того, мы предлагаем механизм градиентной маски с целью скрыть часть обновления весов переобучения, улучшая производительность обобщения нейронной сети.

Список литературы

[1] Zhang, Chiyuan and Bengio, Samy and Hardt, Moritz and Recht, Benjamin and Vinyals, Oriol, “Understanding deep learning (still) requires rethinking generalization”, *Communications of the ACM*, **64** (2019), 107–115.

[2] Sontag, Eduardo D and others, “VC dimension of neural networks”, *NATO ASI Series F Computer and Systems Sciences*, **168** (1998), 69–96.

[3] Bartlett, Peter L and Harvey, Nick and Liaw, Christopher and Mehrabian, Abbas, “Nearly-tight VC-dimension and pseudodimension

bounds for piecewise linear neural networks", *The Journal of Machine Learning Research*, **20** (2019), 2285–2301.

[4] Bartlett, Peter L and Mendelson, Shahar, "Rademacher and Gaussian complexities: Risk bounds and structural results", *Journal of Machine Learning Research*, **3** (2002), 463–482.

[5] Shawe-Taylor, John and Williamson, Robert C, "A PAC analysis of a Bayesian estimator", *Proceedings of the tenth annual conference on Computational learning theory*, 1997, 2–9.

[6] McAllester, David A, "Some pac-bayesian theorems", *Machine Learning*, **37** (1999), 355–363.

[7] Dziugaite, Gintare Karolina and Roy, Daniel M, "Computing nonvacuous generalization bounds for deep (stochastic) neural networks with many more parameters than training data", *arXiv preprint arXiv:1703.11008*, 2017.

[8] Dziugaite, Gintare Karolina and Hsu, Kyle and Gharbieh, Waseem and Arpino, Gabriel and Roy, Daniel, "On the role of data in PAC-Bayes bounds", *International Conference on Artificial Intelligence and Statistics*, 2021, 604–612.

[9] Shorten, Connor and Khoshgoftaar, Taghi M, "A survey on image data augmentation for deep learning", *Journal of big data*, **6** (2019), 1–48.

[10] Hinton, Geoffrey E and Srivastava, Nitish and Krizhevsky, Alex and Sutskever, Ilya and Salakhutdinov, Ruslan R, "Improving neural networks by preventing co-adaptation of feature detectors", *arXiv preprint arXiv:1207.0580*, 2012.

[11] Xie, Zeke and Sato, Issei and Sugiyama, Masashi, "Understanding and scheduling weight decay", *arXiv preprint arXiv:2011.11152*, 2020.

[12] Keskar, Nitish Shirish and Mudigere, Dheevatsa and Nocedal, Jorge and Smelyanskiy, Mikhail and Tang, Ping Tak Peter, "On large-batch training for deep learning: Generalization gap and sharp minima", *arXiv preprint arXiv:1609.04836*, 2016.

[13] Neyshabur, Behnam and Bhojanapalli, Srinadh and McAllester, David and Srebro, Nati, "Exploring generalization in deep learning", *Advances in neural information processing systems*, **30** (2017).

[14] Dinh, Laurent and Pascanu, Razvan and Bengio, Samy and Bengio, Yoshua, "Sharp minima can generalize for deep nets", *International Conference on Machine Learning*, 2017, 1019–1028.

[15] Foret, Pierre and Kleiner, Ariel and Mobahi, Hossein and Neyshabur, Behnam, "Sharpness-aware minimization for efficiently improving generalization", *arXiv preprint arXiv:2010.01412*, 2020.

[16] Park, Hyeyoung and Amari, S-I and Fukumizu, Kenji, "Adaptive natural gradient learning algorithms for various stochastic models", *Neural Networks*, **13** (200), 755–764.

[17] Jastrzębski, Stanisław and Arpit, Devansh and Ballas, Nicolas and Verma, Vikas and Che, Tong and Bengio, Yoshua, "Residual connections encourage iterative inference", *arXiv preprint arXiv:1710.04773*, 2017.

[18] Greff, Klaus and Srivastava, Rupesh K and Schmidhuber, Jürgen, "Highway and residual networks learn unrolled iterative estimation", *arXiv preprint arXiv:1612.07771*, 2016.

[19] Raghu, Maithra and Gilmer, Justin and Yosinski, Jason and Sohl-Dickstein, Jascha, "Svcca: Singular vector canonical correlation analysis for deep learning dynamics and interpretability", *Advances in neural information processing systems*, **30** (2017).

[20] Arpit, Devansh and Jastrzębski, Stanisław and Ballas, Nicolas and Krueger, David and Bengio, Emmanuel and Kanwal, Maxinder S and Maharaj, Tegan and Fischer, Asja and Courville, Aaron and Bengio, Yoshua and others, "A closer look at memorization in deep networks", *International conference on machine learning*, 2017, 233–242.

[21] Martens, James and others, "Deep learning via hessian-free optimization", *ICML*, **27** (2010), 735–742.

[22] Chapelle, Olivier and Erhan, Dumitru and others, "Improved preconditioner for hessian free optimization", *NIPS Workshop on Deep Learning and Unsupervised Feature Learning*, **201** (2011).

[23] He, Kaiming and Zhang, Xiangyu and Ren, Shaoqing and Sun, Jian, "Deep residual learning for image recognition", *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, 770–778.

[24] Krizhevsky, Alex and Hinton, Geoffrey and others, "Learning multiple layers of features from tiny images", 2009.

[25] Deng, Jia and Dong, Wei and Socher, Richard and Li, Li-Jia and Li, Kai and Fei-Fei, Li, "Imagenet: A large-scale hierarchical image database", *2009 IEEE conference on computer vision and pattern recognition*, 2009, 248–255.

[26] Guo, Dongning, "Relative entropy and score function: New information-estimation relationships through arbitrary additive perturbation", *2009 IEEE International Symposium on Information Theory*, 2009, 814–818.

Gradient mask and neural network generalization
Jiang Lei

Within the practical application of neural networks, the number of parameters in the network is much larger than the number of samples in the dataset, however, the network still has good generalization characteristics. Traditionally considered that such over-parameterized and non-convex models can easily fall into local minima while searching for the optimal solution and show low generalization performance, but in fact it is not. Although under some regularization conditions it is possible to effectively control the network generalization error, it is still difficult to explain the generalization problem for large networks. In our work, we determine the difference between the overfitting step and the feature learning step by quantifying the impact of updating one sample during gradient descent on the entire training process, revealing that neural networks generally have less impact on other samples during the overfitting step. In addition, we use the Fisher information matrix to mask the gradient produced by the backpropagation process, thereby slowing down the neural network's overfitting behavior and improving the neural network's generalization performance.

Keywords: Neural Networks, Generalization, Overfitting, Fisher Information.

References

[1] Zhang, Chiyuan and Bengio, Samy and Hardt, Moritz and Recht, Benjamin and Vinyals, Oriol, "Understanding deep learning (still) requires rethinking generalization", *Communications of the ACM*, **64** (2019), 107–115.

[2] Sontag, Eduardo D and others, "VC dimension of neural networks", *NATO ASI Series F Computer and Systems Sciences*, **168** (1998), 69–96.

[3] Bartlett, Peter L and Harvey, Nick and Liaw, Christopher and Mehrabian, Abbas, "Nearly-tight VC-dimension and pseudodimension bounds for piecewise linear neural networks", *The Journal of Machine Learning Research*, **20** (2019), 2285–2301.

[4] Bartlett, Peter L and Mendelson, Shahar, "Rademacher and Gaussian complexities: Risk bounds and structural results", *Journal of Machine Learning Research*, **3** (2002), 463–482.

[5] Shawe-Taylor, John and Williamson, Robert C, "A PAC analysis of a Bayesian estimator", *Proceedings of the tenth annual conference on Computational learning theory*, 1997, 2–9.

[6] McAllester, David A, "Some pac-bayesian theorems", *Machine Learning*, **37** (1999), 355–363.

[7] Dziugaite, Gintare Karolina and Roy, Daniel M, "Computing nonvacuous generalization bounds for deep (stochastic) neural networks with many more parameters than training data", *arXiv preprint arXiv:1703.11008*, 2017.

[8] Dziugaite, Gintare Karolina and Hsu, Kyle and Gharbieh, Waseem and Arpino, Gabriel and Roy, Daniel, "On the role of data in PAC-Bayes bounds", *International Conference on Artificial Intelligence and Statistics*, 2021, 604–612.

[9] Shorten, Connor and Khoshgoftaar, Taghi M, "A survey on image data augmentation for deep learning", *Journal of big data*, **6** (2019), 1–48.

[10] Hinton, Geoffrey E and Srivastava, Nitish and Krizhevsky, Alex and Sutskever, Ilya and Salakhutdinov, Ruslan R, "Improving neural networks by preventing co-adaptation of feature detectors", *arXiv preprint arXiv:1207.0580*, 2012.

[11] Xie, Zeke and Sato, Issei and Sugiyama, Masashi, "Understanding and scheduling weight decay", *arXiv preprint arXiv:2011.11152*, 2020.

[12] Keskar, Nitish Shirish and Mudigere, Dheevatsa and Nocedal, Jorge and Smelyanskiy, Mikhail and Tang, Ping Tak Peter, "On large-batch training for deep learning: Generalization gap and sharp minima", *arXiv preprint arXiv:1609.04836*, 2016.

[13] Neyshabur, Behnam and Bhojanapalli, Srinadh and McAllester, David and Srebro, Nati, "Exploring generalization in deep learning", *Advances in neural information processing systems*, **30** (2017).

[14] Dinh, Laurent and Pascanu, Razvan and Bengio, Samy and Bengio, Yoshua, "Sharp minima can generalize for deep nets", *International Conference on Machine Learning*, 2017, 1019–1028.

[15] Foret, Pierre and Kleiner, Ariel and Mobahi, Hossein and Neyshabur, Behnam, "Sharpness-aware minimization for efficiently improving generalization", *arXiv preprint arXiv:2010.01412*, 2020.

[16] Park, Hyeyoung and Amari, S-I and Fukumizu, Kenji, "Adaptive natural gradient learning algorithms for various stochastic models", *Neural Networks*, **13** (200), 755–764.

[17] Jastrzębski, Stanisław and Arpit, Devansh and Ballas, Nicolas and Verma, Vikas and Che, Tong and Bengio, Yoshua, "Residual connections encourage iterative inference", *arXiv preprint arXiv:1710.04773*, 2017.

[18] Greff, Klaus and Srivastava, Rupesh K and Schmidhuber, Jürgen, "Highway and residual networks learn unrolled iterative estimation", *arXiv preprint arXiv:1612.07771*, 2016.

[19] Raghu, Maithra and Gilmer, Justin and Yosinski, Jason and Sohl-Dickstein, Jascha, "Svcca: Singular vector canonical correlation analysis for deep learning dynamics and interpretability", *Advances in neural information processing systems*, **30** (2017).

[20] Arpit, Devansh and Jastrzębski, Stanisław and Ballas, Nicolas and Krueger, David and Bengio, Emmanuel and Kanwal, Maxinder S and Maharaj, Tegan and Fischer, Asja and Courville, Aaron and Bengio, Yoshua and others, "A closer look at memorization in deep networks", *International conference on machine learning*, 2017, 233–242.

[21] Martens, James and others, "Deep learning via hessian-free optimization", *ICML*, **27** (2010), 735–742.

[22] Chapelle, Olivier and Erhan, Dumitru and others, "Improved preconditioner for hessian free optimization", *NIPS Workshop on Deep Learning and Unsupervised Feature Learning*, **201** (2011).

[23] He, Kaiming and Zhang, Xiangyu and Ren, Shaoqing and Sun, Jian, "Deep residual learning for image recognition", *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, 770–778.

[24] Krizhevsky, Alex and Hinton, Geoffrey and others, "Learning multiple layers of features from tiny images", 2009.

[25] Deng, Jia and Dong, Wei and Socher, Richard and Li, Li-Jia and Li, Kai and Fei-Fei, Li, "Imagenet: A large-scale hierarchical image database", *2009 IEEE conference on computer vision and pattern recognition*, 2009, 248–255.

[26] Guo, Dongning, "Relative entropy and score function: New information-estimation relationships through arbitrary additive perturbation", *2009 IEEE International Symposium on Information Theory*, 2009, 814–818.

Часть 2.
Специальные вопросы теории интеллектуальных систем

Обратная связь в рекуррентных схемах

Н. Ф. Алексиадис[1], В. С. Половников[2] [5], А. А. Часовских[3] [5], В. Г. Шишляков[4]

Показано, что при некоторых естественных ограничениях на элементный базис для любой рекуррентной схемы с фиксированным количеством входов можно построить функционально эквивалентную ей схему в том же базисе с использованием операций суперпозиции и не более чем двукратного применения операции обратной связи при линейном росте числа используемых задержек по сравнению с исходной схемной реализацией. Таким образом, доказана линейность порядка роста памяти при переходе к оптимальной (по количеству задержек) схеме с не более чем двумя обратными связями. В структуре построенных в работе рекуррентных схем с не более чем двумя обратными связями выделяются модули кратковременной и долгосрочной памяти. Полученный результат, в частности, справедлив для класса конечных автоматов, а также для класса нейронных схем, построенных из элементов, содержащих вентили.

Ключевые слова: рекуррентные схемы, обратная связь, линейная оценка, вентили.

[1] *Алексиадис Никос Филиппович* — доцент кафедры прикладной математики и искусственного интеллекта института информационных и вычислительных технологий Национального исследовательского университета «МЭИ», e-mail: aleksiadis@yandex.ru.

Aleksiadis Nikos Philippovich — associate professor, National Research University «MPEI», Instityte of Information and Computation technologies, Chair of applied mathematics and artificial intelligence.

[2] *Половников Владимир Сергеевич* — научный сотрудник кафедры математической теории интеллектуальных систем мех.-мат. ф-та МГУ, e-mail: polovnikov@intsys.msu.ru.

Polovnikov Vladimir Sergeevich — Researcher, Lomonosov Moscow State University, Faculty of Mechanics and Mathematics, Chair of Mathematical Theory of Intellectual Systems.

[3] *Часовских Анатолий Александрович* — доцент кафедры математической теории интеллектуальных систем мех.-мат. ф-та МГУ, e-mail: chasovskikh@mail.ru.

Chasovskikh Anatoly Alexandrovich — associate professor, Lomonosov Moscow State University, Faculty of Mechanics and Mathematics, Chair of Mathematical Theory of Intellectual Systems.

[4] *Шишляков Владимир Геннадьевич* — аспирант кафедры математической теории интеллектуальных систем мех.-мат. ф-та МГУ, e-mail: bolotmaks@yandex.ru.

Shishlyakov Vladimir Gennad'evich — PhD student, Lomonosov Moscow State University, Faculty of Mechanics and Mathematics, Chair of Mathematical Theory of Intellectual Systems.

[5] Исследование выполнено при поддержке Министерства науки и высшего образования Российской Федерации (грант № 075-15-2020-801).

1. Введение и основные понятия

Пусть E — некоторое поле, содержащее, как известно, 0 и 1. Схемы мы будем строить из элементов. Каждый элемент имеет конечное число входов, один выход и функционирует в дискретные моменты времени. В момент времени t, $t \in \{0, 1, 2, ...\}$, на входы элемента поступают значения из E, а на выходе реализуется значение из E. Элемент с n входами называется функциональным, если в любой момент времени t значение его выхода зависит только от значений входов в момент t. Задержкой [1] называется элемент с одним входом и одним выходом, на выходе которого в начальный момент $(t = 0)$ реализуется его начальное состояние $q_0, q_0 \in E$, а в другие моменты времени t на выход поступает значение входа в момент $t - 1$.

Индукцией по построению введем понятие рекуррентной схемы над множеством B, состоящим из функциональных элементов и всех задержек. Каждый элемент из B назовем рекуррентной схемой. Применяя к рекуррентным схемам операции композиции (переименования переменных без отождествления, отождествления переменных, подстановки, разветвления выхода, обратной связи) [1], получаем рекуррентные схемы над B.

Множество всех рекуррентных схем над B обозначим $K(B)$. В дальнейшем предполагаем, что с использованием функциональных элементов из B и операций суперпозиции без обратной связи можно реализовать все линейные функции, а также, что в $K(B)$ содержится вентиль F, то есть схема с двумя входами, реализующая функцию $\phi(x, y)$, для которой выполнены равенства: $\phi(x, 1) = x$ и $\phi(x, 0) = 0$.

Как нетрудно видеть, в этих предположениях при $E = \{0, 1\}$ множество $K(B)$ — все схемы конечных автоматов, а при $E = \mathbb{R}$ и

$$\phi(x, y) = \begin{cases} x, \text{если } y > 0 \\ 0, \text{если } y \leqslant 0 \end{cases}$$

схемами из $K(B)$ можно реализовать все кусочно-линейные нейронные функции, рассматриваемые в работе [2].

Множество схем из $K(B)$, которые можно получить из элементов B при применении не более чем r раз операции обратной связи, обозначим $K_r(B)$. Множество функций, реализуемых схемами из $K(B)$ и $K_r(B)$, мы обозначим Ω и Ω_r, соответственно.

В работе [2] рассмотрен случай нейронных схем, когда $E = \mathbb{R}$, и показано, что в этом случае $\Omega = \Omega_1$.

Используя конструкции из этой работы, нетрудно получить следующее утверждение.

Теорема 1. *При сформулированных выше ограничениях на множество элементов B выполнено: $\Omega = \Omega_2$.*

Из этой теоремы, например, следует, что любой конечный автомат [1] можно реализовать схемой, построенной из элементов, реализующих полное в алгебре логики множество функций и задержек при использовании не более чем двух операций обратной связи.

Сложностью схемы S из $K(B)$ мы называем количество задержек $L(S)$ в этой схеме. Сложностью функции f из класса Ω или Ω_r называем сложность схемы с минимальной сложностью, реализующей эту функцию, и обозначаем $L(f)$ или $L_r(f)$.

Используя конструкцию из работы [2], получаем следующий результат.

Теорема 2. *В рассматриваемых ограничениях на B для заданного n найдется такая константа c, что для любой функции $f, f \in \Omega$, зависящей не более чем от n переменных, выполнено: $L_2(f) \leqslant c \cdot L(f)^2$.*

При этом, как нетрудно видеть, справедливо неравенство: $L(f) \leqslant L_2(f)$.

Отметим, что имеет место улучшение полученной в теореме 2 оценки.

2. Основные результаты

Теорема 3. *В рассматриваемых ограничениях на B для заданного n найдется такая константа c, что для любой функции $f, f \in \Omega$, зависящей не более чем от n переменных, выполнено: $L_2(f) \leqslant c \cdot L(f)$.*

Доказательство. Пусть (1) – канонические уравнения [1] функции f с минимальным числом состояний, где $\overline{q}(t) = (q_1(t), ..., q_k(t)) \in E^k$, $\overline{x}(t) = (x_1(t), ..., x_n(t)) \in E^n$, $y(t) \in E$, $\forall t \in \{0, 1, 2, ...\}$, а $\varphi, \psi \in \Omega_0$.

$$\begin{cases} \overline{q}(0) = (q_1(0), ..., q_k(0)) = (q_1^0, ..., q_k^0) \\ \overline{q}(t+1) = \varphi(\overline{q}(t), \overline{x}(t)) \\ y(t) = \psi(\overline{q}(t), \overline{x}(t)) \end{cases} \tag{1}$$

Согласно [1], каноническим уравнениям (1) соответствует схема S_f из $K(B)$, изображенная на рис. 1. Данную схему будем называть канонической. Задержки с начальными состояниями q_i^0 обозначены на схеме через $З_{q_i^0}$.

Будем называть $\overline{q}(t)$ – вектором состояний схемы, $\overline{x}(t)$ – входами схемы, а $y(t)$ – выходом схемы в момент времени t.

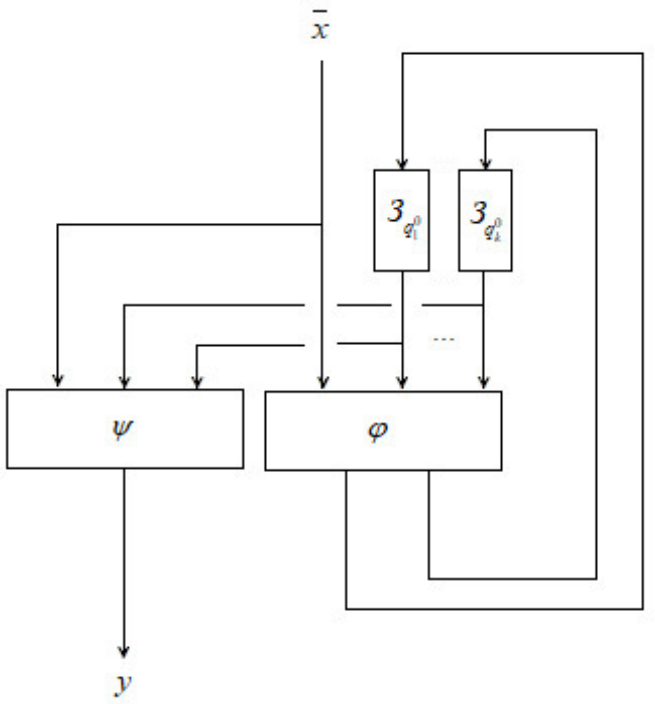


Рис. 1. Каноническая схема $f \in \Omega$

Таким образом, при построении канонической схемы по уравнениям (1) требуется k задержек, которые обозначим $q_1, ..., q_k$, с начальными состояниями $(q_1^0, ..., q_k^0)$ и k операций обратной связи.

Перестроим схему S_f так, чтобы в ней было две обратные связи. Новую схему назовем S_f^2.

Идейно новая схема S_f^2 будет функционировать следующим образом: в определенные моменты времени t' производится фиксация (сохранение) вектора состояний $\overline{q}(t') = (q_1(t'), ..., q_k(t'))$ схемы S_f, затем данный вектор покомпонентно передается через одну из двух обратных связей новой схемы (по одной компоненте за такт).

После передачи компонент вектора состояний через обратную связь они будут аккумулироваться в "верхней части" схемы до тех пор, пока все компоненты вектора с зафиксированными состояниями не передадутся по обратной связи на "верхнюю часть" схемы.

Запоминая входы в моменты времени $t', ..., t' + \Delta - 1$ и используя вектор состояния в момент t', сформированный в "верхней части схемы" получаем новый вектор состояний схемы в момент времени $t' + \Delta$, где Δ – число тактов, которые потребовались для передачи всех компонент

вектора состояний $(q_1(t'), ..., q_k(t'))$ через обратную связь и аккумулирования их на "верхней части" схемы.

Далее производится обновление состояния задержек, которые изначально передавали значения $(q_1(t'), ..., q_k(t'))$ на обратную связь. При обновлении состояний на задержки помещаются новые значения $(q_1(t' + \Delta), ..., q_k(t' + \Delta))$. И со следующего такта описанная процедура покомпонентной передачи вектора состояний повторяется.

Таким образом, получается, что раз в Δ тактов в схеме S_f^2 на фиксированных соединениях (проводах) возникают значения, совпадающие со значениями состояний в канонической схеме S_f. Поэтому, подсоединив данные "провода" и вход $\overline{x}(t)$ к соответствующим входам функции ψ, получим, что раз в Δ тактов можно реализовать правильный выход $y(t)$.

Однако, конечной целью является получение на схеме S_f^2 выхода $y(t)$, совпадающего с выходом схемы S_f в любой момент времени $t, t \in \{0, 1, 2, ...\}$. Для этого во всех тактах $t = t' + i, i \in \{1, ..., \Delta - 1\}$ будем вычислять значение $y(t)$ в каждый момент времени t, используя состояния $(q_1(t'), ..., q_k(t'))$ в момент времени t' и входы $\overline{x}(t), ..., \overline{x}(t - i) = \overline{x}(t')$ при помощи функций ψ^i [1].

Теперь перейдем от идеи к подробному доказательству теоремы. Разобьем схему S_f^2 на крупные блоки (модули) и рассмотрим общую структуру схемы (см. рис. 2). После чего перейдем к описанию каждого блока с доказательством того, что он работает корректно.

В схеме на рис. 2 тонкими линиями обозначаются связи ширины 1 (передача однокомпонентного вектора, или, что эквивалентно, элемента поля E), более широкими линиями обозначаются связи с передачей многокомпонентных векторов в каждый конкретный момент времени.

Каждый блок имеет имя, а рядом с выходом каждого блока и рядом со входом схемы находится описание сигналов, которые появляются на выходе данного блока в каждый момент времени. Моменты времени представлены в виде $t = k \cdot s + i$, где k – размерность вектора состояний $(q_1(t), ..., q_k(t))$ в канонической схеме, $s \in \{0, 1, 2, ...\}$, а $i \in \{0, 1, 2, ..., k - 1\}$. Моменты времени $k \cdot s + i$, когда s фиксировано, будем называть s-ой эпохой обновления состояния (или просто, эпохой). Также, чтобы подчеркнуть, что в основной логике значение какого-то компонента описываемого выхода не влияет на отображение, осуществляемое схемой, будем использовать обозначение "*" . Данное выражение означает, что на месте, в котором оно используется, может стоять произвольный элемент поля E.

Обозначения выходов блоков унифицированы и имеют вид $\overline{y}_B$, где B - имя блока на схеме. Если потребуется взять i-ую компоненту вы-

хода блока B в момент времени t, то будем использовать выражение $\overline{y}_B^{(i)}(t)$. Если требуется взять несколько компонент $i_1, ..., i_l$ блока, то будем использовать выражение $\overline{y}_B^{(i_1),...,(i_l)}(t)$, полагая, по определению, что это вектор $(\overline{y}_B^{(i_1)}(t), ..., \overline{y}_B^{(i_l)}(t))$.

Под обозначением $\delta_{i,j}$ на рис. 2 понимается символ Кронекера $\delta_{i,j} = \begin{cases} 1, i = j \\ 0, i \neq j \end{cases}$.

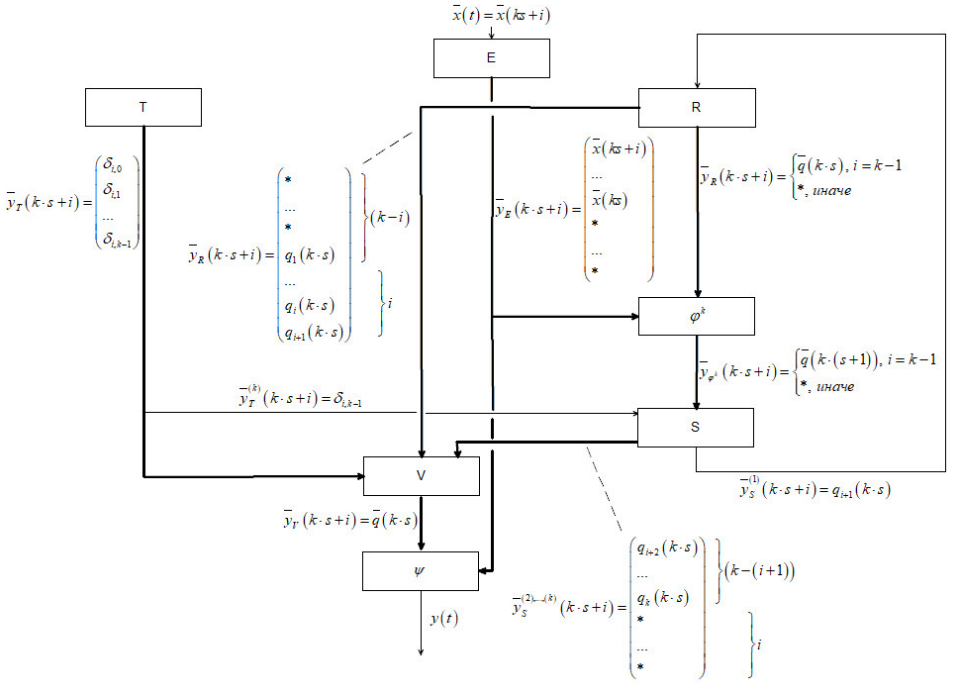


Рис. 2. Схема с двумя обратными связями и разбиением на блоки

Отметим, что, если каждый блок в S_f^2 реализуется некоторой схемой, то S_f^2 – тоже схема, так как вся схема может быть построена из этих блоков с использованием операции подстановки, обратной связи и, в некоторых случаях (блоки φ^k, S), отождествления переменных.

Далее подробнее рассмотрим блоки схемы, изображенной на рис. 2.

• **Блок T**. Данный блок является таймером и возвращает в каждый момент времени k-мерный вектор $(\overline{y}_T^{(1)}, ..., \overline{y}_T^{(k)}) = (\tau_0(t), ..., \tau_{k-1}(t)) = (\delta_{i,0}, \delta_{i,1}, ..., \delta_{i,k-1})$, в котором при любом $t = k \cdot s + i$ содержится един-

ственная единица, стоящая на месте $\tau_i(t)$ при любом $s = 0, 1, ...$ (в любой эпохе).

Таким образом, данный блок реализует счетчик итераций внутри одной эпохи. В данной теореме, чтобы не усложнять и не запутывать доказательство, реализуем его при помощи схемы с одной обратной связью и k задержками (см. рис. 3).

Мы не производим оптимизацию числа задержек в данном блоке, так как нам важна оценка по порядку числа задержек в схеме S_f^2. Данная реализация не портит порядок оценки, а оптимизация числа задержек в данном блоке не повлияет на порядок итоговой оценки. Хотя следует отметить, что данный блок может быть оптимизирован. Даже в общем случае можно, очевидно, обойтись $\lceil log_2(k) \rceil$ задержками, а в частном случае, когда $E = \mathbb{R}$ можно обойтись реализацией данного блока с нулевым числом задержек и без обратных связей взамен увеличения размерности вектора состояний на 1 в канонической схеме [2].

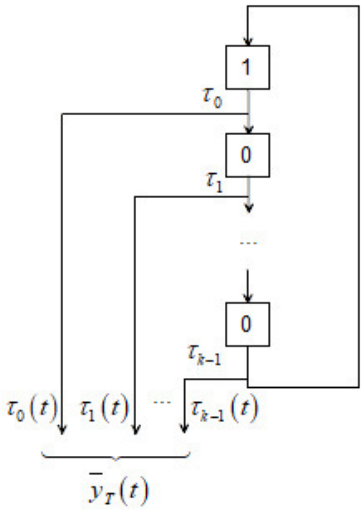


Рис. 3. Реализация схемы блока-таймера **T**. Задержки обозначены квадратами, внутри квадрата обозначено начальное состояние задержки

Очевидно, что

$$\overline{y}_T(k \cdot s + i) = (\tau_0(k \cdot s + i), ..., \tau_{k-1}(k \cdot s + i)) = (\delta_{i,0}, \delta_{i,1}, ..., \delta_{i,k-1}) \quad (2)$$

Другими словами, при $i = 0, 1, ..., k-1$ выходом схемы, реализующей блок **T**, является вектор из E^k с единственной ненулевой компонентой $\tau_i(k \cdot s + i)$ и нулями на остальных компонентах, то есть $\overline{y}_T(k \cdot s + i) = (0, ..., 0, \underbrace{1}_{i \text{ позиция}}, 0, ..., 0)$.

• **Блок E**. Блок **E**, который мы будем также называть блоком кратковременной памяти, представляет собой линию задержек с отводами [3], сохраняющим все входы, пришедшие на схему с начала эпохи. Выходы данного блока передаются на функции $\varphi^j, \psi^j,\ j \in \{0, 1, ..., k-1\}$, используемые далее в схеме S_f^2.

Говоря формально, на вход блока **E** приходит значение $\overline{x}(t) = (x_1(t), ..., x_n(t))$, а на выходе требуется получить значения $\overline{x}(t), ..., \overline{x}(t-(k-1))$.

Для реализации блока **E** рассмотрим вспомогательные блоки $E_j, j \in 1, ..., n$, схемы которых изображены на рис. 4.

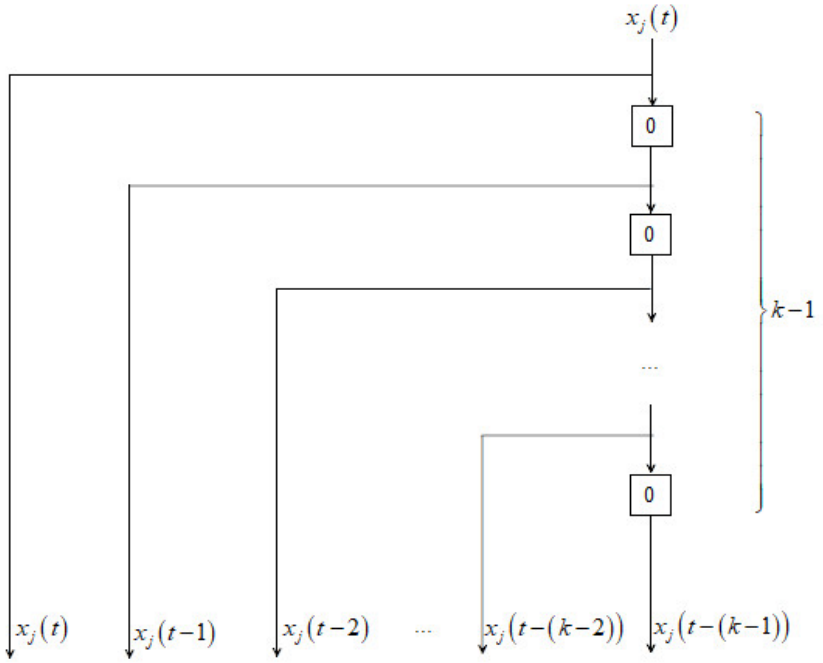


Рис. 4. Реализация схемами блоков $E_j, j \in 1, ..., n$

Очевидно, что каждый блок E_j принимает на вход значение $x_j(t)$, являющееся j-ой компонентой вектора $\overline{x}(t)$, и выдает на выходах $x_j(t), x_j(t-1), ..., x_j(t-(k-1))$ при $t \geqslant k-1$. Также очевидно, что при $t < k-1$ блоком E_j возвращаются значения $\underbrace{x_j(t), ..., x_j(1), x_j(0)}_{t+1}, \underbrace{0, ..., 0}_{k-t-1}$.

Но тогда, представляя t, как $t = k \cdot s + i$, получаем, что каждый блок E_j в момент времени t возвращает значения $x_j(k \cdot s + i), ..., x_j(k \cdot s + i -$

$(k-1))$ при $t \geqslant k-1$ и значения $\underbrace{x_j(i),...,x_j(1),x_j(0)}_{i+1},\underbrace{0,...,0}_{k-i-1}$ в случае $t < k-1$.

Выделяя в случае $t \geqslant k-1$ первые $i+1$ компонент вектора $x_j(k\cdot s+i),...,x_j(k\cdot s+i-(k-1))$, получаем при любом $t = k\cdot s+i$ вектор $\underbrace{x_j(k\cdot s+i),...,x_j(k\cdot s)}_{i+1},\underbrace{*,...,*}_{k-i-1} = \underbrace{x_j(k\cdot s+i),...,x_j(k\cdot s)}_{i+1},\underbrace{*,...,*}_{k-i-1}$ на выходе блока E_j.

Схемы блоков $E_j, j \in \{1,...,n\}$ будем сокращенно обозначать так, как это сделано на рис. 5.

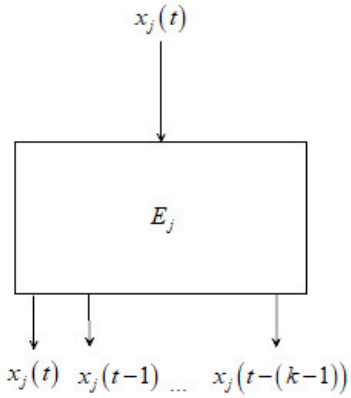


Рис. 5. Обозначение схем блоков $E_j, j \in 1,...,n$

Из блоков $E_j, j \in \{1,...,n\}$ составим схему блока **E**. Схема данного блока изображена на рис. 6.

Но тогда, очевидно, что блок **E** в каждый момент времени $t = k\cdot s+i$ возвращает значения $\underbrace{\overline{x}(k\cdot s+i),...,\overline{x}(k\cdot s)}_{i+1},\underbrace{*,...,*}_{k-i-1}$.

Также отметим, что при построении блока **E** используется $n(k-1)$ задержек.

• **Блок сжатия S.** Задачей данного блока является сжатие вектора состояний до одной компоненты для дальнейшей передачи данных компонент через единственную обратную связь (по одной компоненте за такт).

При этом, в определенные моменты времени (в моменты $t = k\cdot(s+1), s = 0,1,2,...$), то есть при окончании эпохи, требуется уметь пересчитывать хранящиеся и циркулирующие по блокам **S**,**R**,φ^k состояния

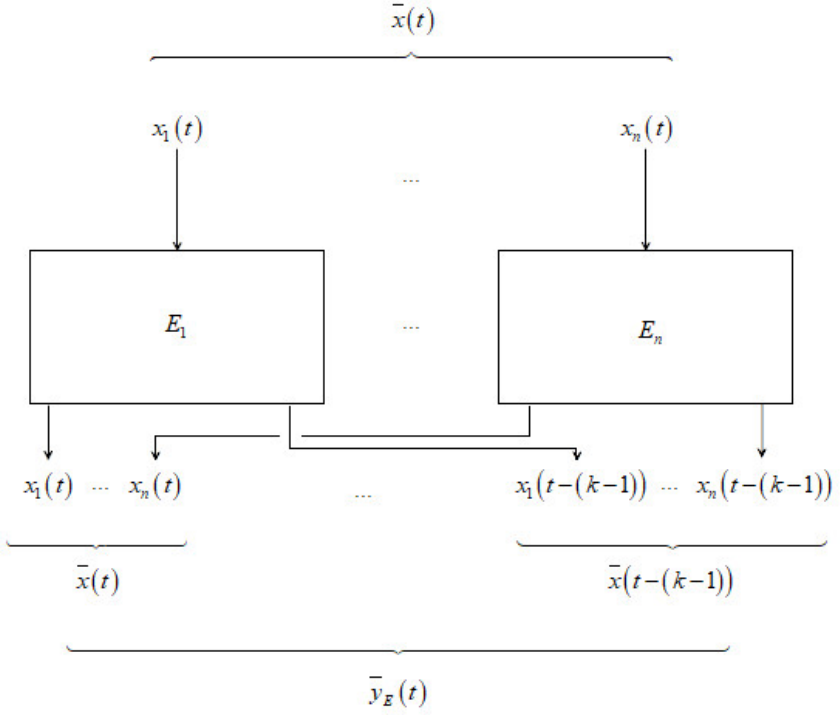


Рис. 6. Схема блока **E**

$\overline{q}(k \cdot s)$ в состояния $\overline{q}(k \cdot (s+1))$ и обновлять их в блоке **S** для того, чтобы в дальнейшем уже новые состояния циркулировали по контуру **S**,**R**,φ^k.

Для реализации схемы блока **S** потребуется дополнительный элемент, который в дальнейшем будем называть блоком памяти (см. рис. 7 (а, б)). В данном блоке расположены три входа τ', z, q и один выход y_m.

Суть данного блока состоит в том, что в зависимости от того, какое из значений $\tau' \in \{0,1\}$ придет на вход, в состояние задержки, находящейся внутри данного блока, будет записано либо значение z (при $\tau' = 1$), либо значение q (при $\tau' = 0$).

Другими словами, блок памяти реализует следующую функцию: $y_m(t+1) = m(\tau'(t), z(t), q(t)) = \begin{cases} z(t), \text{если } \tau'(t) = 1, t \geqslant 1 \\ q(t), \text{если } \tau'(t) = 0, t \geqslant 1 \\ a_0,\ t = 0 \end{cases}$, где $a_0 \in E$ – начальное состояние задержки в момент времени $t = 0$.

Очевидно, что условный оператор, действующий по принципу $y = \begin{cases} z,\ \tau' = 1 \\ q,\ \tau' = 0 \end{cases}$ можно реализовать следующей формулой: $\phi(z, \tau') + \phi(q, 1 -$

τ') (предполагается, что $\tau' \in \{0,1\}$). Но тогда, для добавления временно́го лага, требуется добавить задержку. Таким образом, требуемую формулу можно представить в виде формулы (3).

$$\mathcal{Z}_{a_0}(\phi(z,\tau') + \phi(q, 1-\tau')) \tag{3}$$

Представив формулу (3) в виде схемы, получаем рис. 7 (б), на котором изображена схема блока памяти. На рис. 7 (а) изображено сокращенное обозначение данного блока при построении схем.

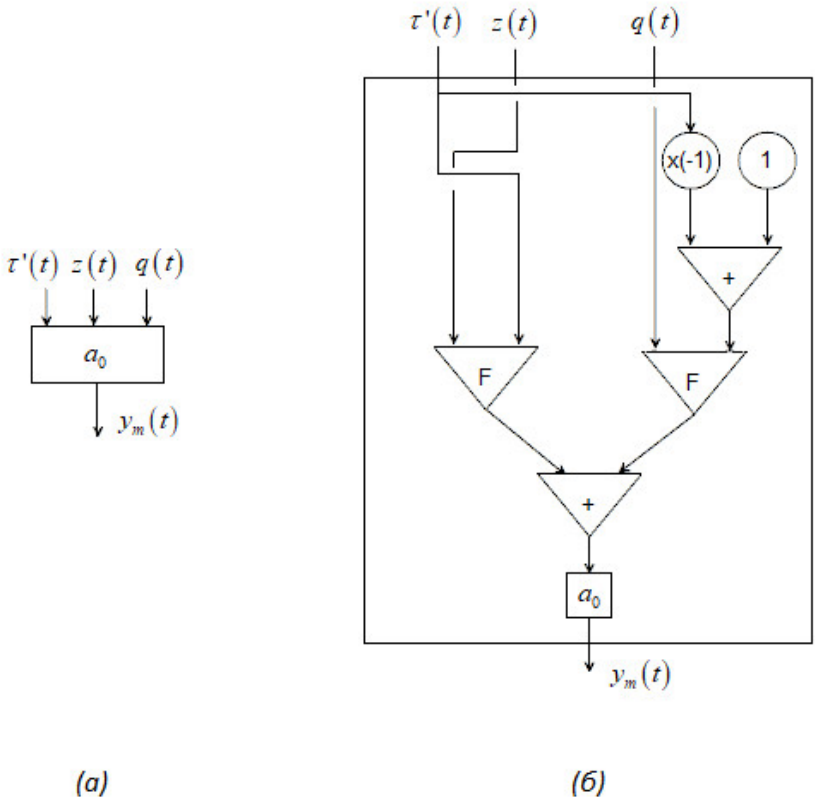


Рис. 7. Блок памяти

Учитывая схемное обозначение блока памяти, введенное на рис. 7 (а), рассмотрим схему, реализующую блок сжатия **S** (см. рис. 8).

Покажем, что данная схема действительно реализует нужную нам функцию $\overline{y}_S(k\cdot s+i) = (\overline{y}_S^{(1)}(k\cdot s+i), ..., \overline{y}_S^{(k)}(k\cdot s+i)) = (q_{i+1}(k\cdot s), q_{i+2}(k\cdot s), ..., q_k(k\cdot s), *, ..., *)$, в предположении, что подключенные к ней блоки выдают такие выходы, как на схеме, изображенной на рис. 2.

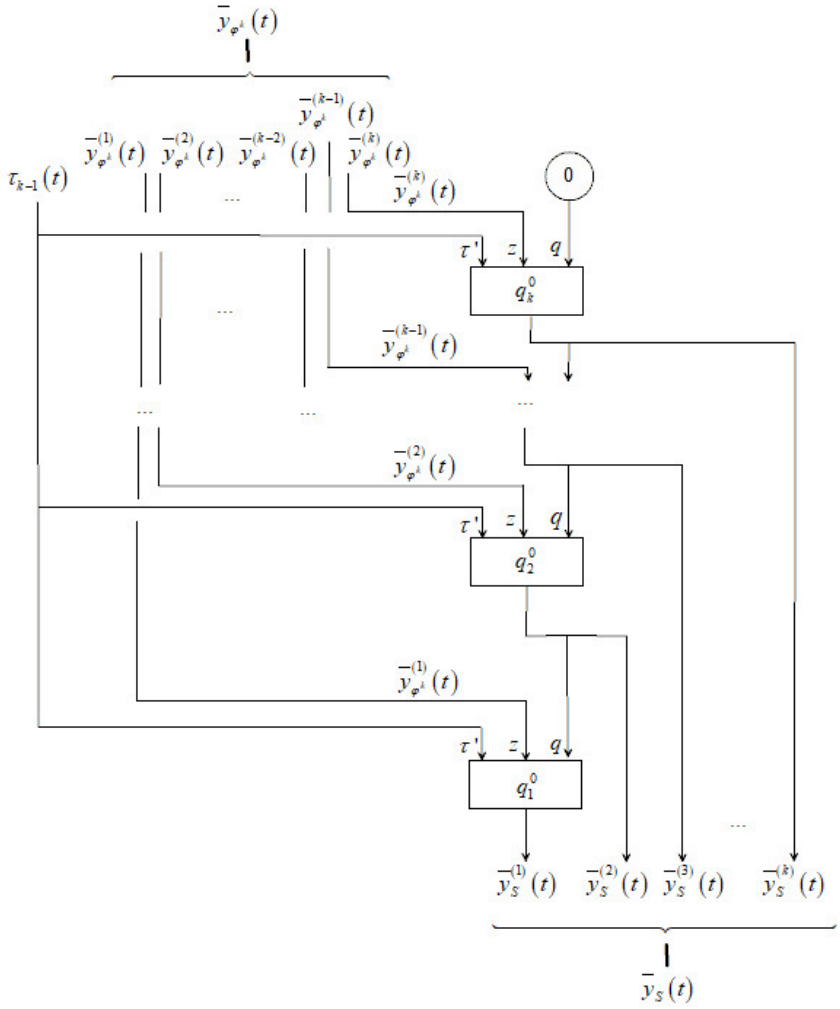


Рис. 8. Реализация блока сжатия **S**

В схеме на рис. 8 вход $\tau_{k-1}(t)$ является последней компонентой выхода $\overline{y}_T(t) = (\tau_0(t), \tau_1(t), ..., \tau_{k-1}(t))$ блока **T** и, как говорилось ранее, если представить время t в виде разложения $t = k \cdot s + i$, то $\tau_{k-1}(k \cdot s + i) = 1 \iff \delta_{i,k-1} = 1 \iff i = k - 1$.

Таким образом, в моменты времени $t = k \cdot s + i$ при $i = k - 1$ на управляющие входы τ' всех блоков памяти в схеме блока **S** подаются значения $\tau' = 1$.

Обозначим блоки памяти, имеющие начальные значения $q_1^0, ..., q_k^0$ на схеме из рис. 7, как $q_1, ..., q_k$, соответственно.

Отсюда следует, что в блоки памяти $q_1, ..., q_k$ в данный момент времени $t = k \cdot s + i, i = k - 1$ будут записаны значения со входов типа z на данных блоках, которые совпадают с $\overline{y}^{(1)}_{\varphi^k}(k \cdot s + i), ..., \overline{y}^{(k-1)}_{\varphi^k}(k \cdot s + i), \overline{y}^{(k)}_{\varphi^k}(k \cdot s + i)$, соответственно.

При этом, согласно общей идее, блок φ^k возвращает в рассматриваемый момент времени $t = k \cdot s + i$ при $i = k - 1$ значения $\overline{q}(k \cdot (s + 1)) = (q_1(k \cdot (s + 1)), ..., q_k(k \cdot (s + 1)))$, где $q_j(k \cdot (s + 1)), j = 1, ..., k$ – значения состояний в момент времени $k \cdot (s+1)$ у канонической схемы рассматриваемой функции f. Поэтому $\overline{y}_{\varphi^k}(k \cdot s + i) = \overline{q}(k \cdot (s + 1))$ при $i = k - 1$.

Из сказанного вытекает, что в моменты времени $t = k \cdot s + i, i = k - 1$ на задержках в блоках памяти $q_1, ..., q_k$ будут храниться значения $q_1(k \cdot (s + 1)), ..., q_k(k \cdot (s+1))$, соответственно. При этом, согласно устройству блоков памяти, в момент времени $k \cdot (s + 1)$ блоки $q_1, ..., q_k$ выдадут значения, хранящиеся на задержках, то есть значения $q_1(k \cdot (s+1)), ..., q_k(k \cdot (s+1))$, соответственно.

Учитывая, что блоки памяти $q_1, ..., q_k$ в начальный момент времени $t = 0$ инициализированы корректными значениями (совпадающими со значениями компонент состояний в исходной схеме в момент времени $t = 0$), получаем, что при $t = 0$ блоки памяти подадут на выход корректные значения $(q^0_1, ..., q^0_k) = (q_1(0), ..., q_k(0))$.

Таким образом, показано, что в моменты времени $t = k \cdot s$ блоки $q_1, ..., q_k$ выдают значения $q_1(k \cdot s), ..., q_k(k \cdot s)$, соответственно. При этом на обратную связь схемы S^2_f идет только выход блока q_1, поэтому в моменты времени $t = k \cdot s$ на обратную связь будет идти значение $q_1(k \cdot s)$, которое совпадает, по определению, со значением $q(k \cdot s)$ в каждый из моментов времени $t = k \cdot s$.

Отметим, что на блок $\mathbf{V}$ от блока $\mathbf{S}$ идут все оставшиеся выходы $\overline{y}^{(2)}_S(t), ..., \overline{y}^{(k)}_S(t)$ и в момент времени $t = k \cdot s$ данные значения совпадают с вектором $(q_2(k \cdot s), ..., q_k(k \cdot s)) = (q_{i+2}(k \cdot s), ..., q_{i+k}(k \cdot s))$, который, очевидно, укладывается (принадлежит) в требуемую на рис. 2 последовательность значений $(q_{i+2}(k \cdot s), ..., q_{i+k}(k \cdot s), *, ..., *)$.

Покажем, что во все остальные моменты времени схема продолжит выдавать корректные результаты.

Действительно, в моменты времени $t = k \cdot s + i,\ i \in \{1, ..., k - 1\}$ все блоки памяти будут выдавать значения, хранящиеся в их задержках в моменты времени $t = k \cdot s + i,\ i \in \{0, ..., k - 2\}$. Но в рассматриваемые моменты времени значение $\tau_{k-1}(t) = 0$, поэтому в состояния задержек на блоках памяти будут записываться входы типа q, которые подключены к выходам предыдущих блоков памяти для блоков $q_1, ..., q_{k-1}$ и генератору нуля для блока q_k. Так как 0 - линейная функция, то она, очевидно, представима в виде схемы без обратных связей.

Отсюда немедленно вытекает, что в моменты $t = k \cdot s + i,\ i \in \{1, ..., k-1\}$ на выходе блока q_1 будут появляться значения, реализуемые на выходах блоков $q_{1+i} = q_{i+1}$ в момент времени $t = k \cdot s$. Но в момент времени $t = k \cdot s$ на выходе каждого блока $q_j,\ j = 1, ..., k$ реализуются значения $q_j(k \cdot s),\ j = 1, ..., k$, соответственно.

Таким образом, в моменты времени $t = k \cdot s + i,\ i \in \{1, ..., k-1\}$ на выходе блока q_1 будут появляться значения $q_{i+1}(k \cdot s)$, что доказывает корректность работы выхода, идущего на обратную связь, у описанной на рис. 8 схемы блока **S**.

Рассмотрим, что будет происходить с остальными выходами $\overline{y}_S^{(2)}(k \cdot s+i), ..., \overline{y}_S^{(k)}(k \cdot s+i)$ блока в моменты времени $t = k \cdot s+i,\ i \in \{1, ..., k-1\}$. По аналогии с блоком q_1, на выходах остальных блоков $q_b,\ b \in \{2, ..., k\}$ будут появляться значения, стоящие на блоках $q_{b+i} = q_{i+b}$ в момент времени $t = k \cdot s$, если $i + b \leqslant k$. В случае же $i + b > k$ выходы на блоках q_{i+b}, очевидно, будут нулевыми.

Так как в момент времени $t = k \cdot s$ блоки q_{i+b} выдают значения $q_{i+b}(k \cdot s)$, то из сказанного выше получаем, что блоки q_b выдают значения $\begin{cases} q_{i+b}(k \cdot s),\ i + b \leqslant k \\ 0,\ i + b > k \end{cases}$, в моменты времени $t = k \cdot s + i,\ i \in \{1, ..., k-1\}$. Последнее, очевидно, укладывается в последовательность $(\overline{y}_S^{(2)}(k \cdot s + i), ..., \overline{y}_S^{(k)}(k \cdot s + i)) = (q_{i+2}(k \cdot s), ..., q_k(k \cdot s), *, ..., *)$.

Причем отметим, что на выход данного блока, подсоединенного к обратной связи, идет всегда одно значение, поэтому доказано, что обратная связь на рис. 2 единственная (вторая обратная связь спрятана в блоке **T** и, как мы увидим далее, ни в одном другом блоке обратных связей нет).

Также в дальнейшем пригодится знание о числе задержек в блоке **S**. Очевидно, что их столько же, сколько блоков памяти в схеме на рис. 8, то есть k штук.

• **Блок распараллеливания R.** Данный блок требуется для того, чтобы принимать состояния, переданные блоком сжатия через единственную обратную связь на укрупненной схеме, изображенной на рис. 2, сохранять их, а затем, в определенный момент времени, возвращать весь "перетекший" через обратную связь вектор $\overline{q}(k \cdot s)$ и передавать его в блок φ^k для вычисления нового состояния схемы.

Представим схему, реализующую этот блок. Данная схема изображена на рис. 9. На вход данной схемы поступает выход $\overline{y}_S^{(1)}(k \cdot s + i)$ схемы **S**, переданный через обратную связь, который на рис. 9 обозначен, как $q(t)$.

На выходе схемы при $t \geqslant k-1$ возникают значения $q(t), q(t-1), ..., q(t-(k-1))$ в силу того, что в схеме используется $k-1$ задержка. Данные задержки названы $q_{k+1}, ..., q_{2k-1}$ (см. рис. 9).

В целом, не имеет значения, какими начальными состояниями инициированы задержки, изображенные на схеме, поэтому, без ограничения общности, будем считать, что они инициализированы нулями поля E.

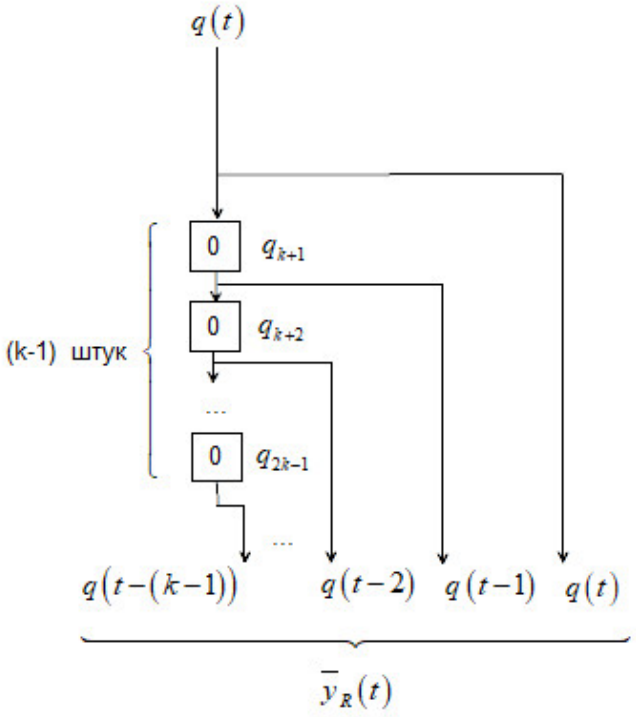


Рис. 9. Реализация блока распараллеливания **R**

Покажем теперь, что схема, изображенная на рис. 9 действительно реализует требуемую функцию $\overline{y}_R(k \cdot s + i) = (*, ..., *, q_1(k \cdot s), ..., q_i(k \cdot s), q_{i+1}(k \cdot s))$ в предположении, что подключенные к ее входам блоки выдают такие выходы, как на схеме, изображенной на рис. 2.

Действительно, по ранее показанному, в каждый момент времени $t = k \cdot s + i$ по обратной связи передается единственное значение $q_{i+1}(k \cdot s)$, где $\overline{q}(k \cdot s) = (q_1(k \cdot s), ..., q_k(k \cdot s))$ – корректное значение вектора состояний канонической схемы в момент времени $t = k \cdot s$.

Но тогда в момент времени при $i = 0$ получаем, что $q(t) = q_{i+1}(k \cdot s) = q_1(k \cdot s)$. При этом состояние задержки q_{k+1} при $i = 0$ станет равным $q_{i+1}(k \cdot s) = q_1(k \cdot s)$, поэтому на следующем такте, при $i = 1$, $q(t)$ станет

равным $q_{i+1}(k\cdot s) = q_2(k\cdot s)$, а $q(t-1)$ станет равным состоянию, записанному в задержку q_{k+1} в предыдущий момент времени, то есть значению $q_1(k\cdot s)$.

Поступая аналогично ("проталкивая вниз" по задержкам блока $\mathbf{R}$ все ранее поступившие состояния), получаем, что при каждом $i \in \{1, ..., k-1\}$ данный блок будет выдавать набор значений следующего вида: $\overline{y}_R(k\cdot s+i) = (\overline{y}_R^{(1)}(k\cdot s+i), ..., \overline{y}_R^{(k)}(k\cdot s+i)) = (q(t-(k-1)), ..., q(t)) = (*, ..., *, q_1(k\cdot s), ..., q_i(k\cdot s), q_{i+1}(k\cdot s))$.

При этом, в моменты времени при $i = k-1$ на выходах $q(t-(k-1)), ..., q(t)$ появятся значения $q_1(k\cdot s), ..., q_{i+1}(k\cdot s) = q_1(k\cdot s), ..., q_k(k\cdot s)$, соответственно (так как в схеме на рис. 9 ровно k выходов, то все они получаются заполненными).

Другими словами, при $i = k-1$ имеем $\overline{y}_R(k\cdot s+i) = (q_1(k\cdot s), ..., q_k(k\cdot s)) = \overline{q}(k\cdot s)$.

Таким образом, схема, изображенная на рис. 9, реализует правильную функцию $\overline{y}_R(k\cdot s+i)$ (обладающую требуемыми свойствами на обоих выходах блока $\mathbf{R}$ в схеме, изображенной на рис. 2).

Отметим также, что в данном блоке была использована $k-1$ задержка.

• Блок φ^k. В данном блоке реализуется единственная функция $\varphi^k(\overline{q}(t-(k-1)), \overline{x}(t-(k-1)), ..., \overline{x}(t))$, которая определяется по рекуррентной формуле (4).

$$\begin{cases} \varphi^k(\overline{q}(t-(k-1)), \overbrace{\overline{x}(t-(k-1)), ..., \overline{x}(t)}^{k}) = \\ \quad = \varphi(\varphi^{k-1}(\overline{q}(t-(k-1)), \overline{x}(t-(k-1)), ..., \overline{x}(t-1)), \overline{x}(t)) \\ \varphi^1(\overline{q}(t), \overline{x}(t)) = \varphi(\overline{q}(t), \overline{x}(t)) \end{cases} \tag{4}$$

В силу того, что $\varphi(\overline{q}, \overline{x}) \in \Omega_0$, а k – конечно, то $\varphi^k \in \Omega_0$, так как, очевидно, что схема функции φ^k является суперпозицией схем для функции $\varphi(\overline{q}, \overline{x})$.

Покажем, что схема функции φ^k, которую будем называть блоком φ^k, будет выдавать корректный выход $\overline{y}_{\varphi^k}(k\cdot s+i) = \begin{cases} \overline{q}(k\cdot(s+1)),\ i = k-1 \\ *,\ \text{иначе} \end{cases}$ в любой момент времени $t = k\cdot s+i$.

Отметим сначала, что, по определению, функция φ^k возвращает по набору значений $\overline{q}(t-(k-1)), \overline{x}(t-(k-1)), \overline{x}(t)$, значение $\overline{q}(t+1)$ [1].

Согласно рис. 2, входы блока φ^k подключены к блокам $\mathbf{R}$, $\mathbf{E}$, соответственно. Напомним, что блок $\mathbf{E}$ возвращает в момент времени $t = k\cdot s+i,\ i = k-1$ значения $\overline{x}(t), \overline{x}(t-1), ..., \overline{x}(t-(k-1))$ $\overset{t=k\cdot s+(k-1)}{=}$

$\overline{x}(k \cdot s + (k-1)), \overline{x}(k \cdot s + (k-2)), ..., \overline{x}(k \cdot s)$, а блок $\mathbf{R}$ возвращает в указанный момент времени t значения $\overline{q}(k \cdot s)$.

Подключим выходы $\overline{x}(k \cdot s + (k-1)), \overline{x}(k \cdot s + (k-2)), ..., \overline{x}(k \cdot s)$ блока $\mathbf{E}$ к блоку φ^k в нужном (обратном) порядке $\overline{x}(k \cdot s), ..., \overline{x}(k \cdot s + (k-2)), \overline{x}(k \cdot s + (k-1))$.

Тогда имеем $\varphi^k(\overline{q}(k \cdot s), \overline{x}(k \cdot s), ..., \overline{x}(k \cdot s + (k-2)), \overline{x}(k \cdot s + (k-1))) \stackrel{def}{=} \overline{q}(k \cdot s + (k-1) + 1) = \overline{q}(k \cdot s + k) = \overline{q}(k \cdot (s+1))$, откуда и следует доказываемое.

Отметим также, что при построении блока φ^k задержки не использовались.

• Блок $\mathbf{V}$. Можно заметить, что в каждый момент времени $t = k \cdot s + i$ на выходах блоков $\mathbf{S}$ и $\mathbf{R}$ всегда можно найти все значения $q_j(k \cdot s),\ j = 1, ..., k$. Однако при изменении $i \in \{0, ..., k-1\}$ значения $q_j(k \cdot s),\ j = 1, ..., k$ постоянно сдвигаются (то есть с изменением времени t начинают выдаваться с других выходов блоков $\mathbf{S}$, $\mathbf{R}$).

Сделаем так, чтобы вектор значений $q_j(k \cdot s),\ j = 1, ..., k$ выдавался в каждый момент времени с одних и тех же "проводов" , то есть чтобы в схеме были "провода" , дающие стабильный сигнал $(q_1(k \cdot s), ..., q_k(k \cdot s))$ (без искажений и сдвигов).

Эту задачу выполняет блок $\mathbf{V}$. Формально, данный блок принимает почти все выходы блоков $\mathbf{S}$, $\mathbf{R}$ (почти все, а не все, так как выходы данных блоков перекрываются (дублируются) в значении $q_{i+1}(k \cdot s)$), содержащие "плавающие" значения $q_j(k \cdot s),\ j = 1, ..., k$ и выдает стабильное внутри каждой эпохи значение $\overline{y}_V(k \cdot s + i) = \overline{q}(k \cdot s)$.

Отметим, что на вход блока $\mathbf{V}$ передаются также сигналы с блока $\mathbf{T}$. Данная мера является вынужденной и чисто технической.

Рассмотрим идею построения схемы блока $\mathbf{V}$. Для начала, обозначим $w_2(t), ..., w_k(t)$ выходы блоков памяти $q_2, ..., q_k$ из блока $\mathbf{S}$ в момент времени t, а $w_{k+1}(t), ..., w_{2k-1}(t)$ – выходы задержек $q_{k+1}, ..., q_{2k-1}$ из блока $\mathbf{R}$ в момент времени t. Под $w_1(t)$ будем понимать выход блока q_1. Данный выход, формально, принадлежит блокам $\mathbf{S}$, $\mathbf{R}$ одновременно. Но мы будем полагать, что $w_1(t)$ – выход блока $\mathbf{R}$, что соответствует фактическим свободным входам и выходам блоков $\mathbf{S}$, $\mathbf{R}$ (выход $w_1(t)$ в блоке $\mathbf{S}$ подключен по обратной связи ко входу блока $\mathbf{R}$, поэтому, фактически, он не является свободным выходом блока $\mathbf{S}$).

Исходя из доказанных свойств сигналов на выходах в блоках $\mathbf{S}$, $\mathbf{R}$ следует, что при $t = k \cdot s + i$ выполняется формула (5).

$$(w_{2k-1-i+1}(k\cdot s+i),...,w_{2k-1}(k\cdot s+i),$$
$$w_1(k\cdot s+i),w_2(k\cdot s+i),...,w_{k-i}(k\cdot s+i))=$$
$$=(q_1(k\cdot s),...,q_k(k\cdot s)) \quad (5)$$

В формуле (5) полагается, что, по определению, $w_i(t),...,w_j(t) = \begin{cases} \varnothing,\ j<i, \\ w_j(t),\ i=j, \\ w_i(t),...,w_j(t),\ j>i. \end{cases}$.

То есть, другими словами, при $i=0$ формула (5) принимает вид $(w_1(k\cdot s+i),w_2(k\cdot s+i),...,w_k(k\cdot s+i))=(q_1(k\cdot s),...,q_k(k\cdot s))$, а при $i=k-1$ получаем $(w_{2k-1-(k-1)+1}(k\cdot s+i),...,w_{2k-1}(k\cdot s+i),w_1(k\cdot s+i),...,w_{k-(k-1)}(k\cdot s+i))=(q_1(k\cdot s),...,q_k(k\cdot s))$, то есть $(w_{k+1}(k\cdot s+i),...,w_{2k-1}(k\cdot s+i),w_1(k\cdot s+i))=(q_1(k\cdot s),...,q_k(k\cdot s))$.

Далее, рассмотрим блоки, составленные из элементов F, соответствующих функциям $\phi(x,y)$, и построенные над скользящими окнами $(w_{2k-1-i+1}(t),...,w_{2k-1}(t),w_1(t),w_2(t),...,w_{k-i}(t))$. Данные блоки изображены на рис. 10. Такие блоки будем называть еще χ-многополюсниками.

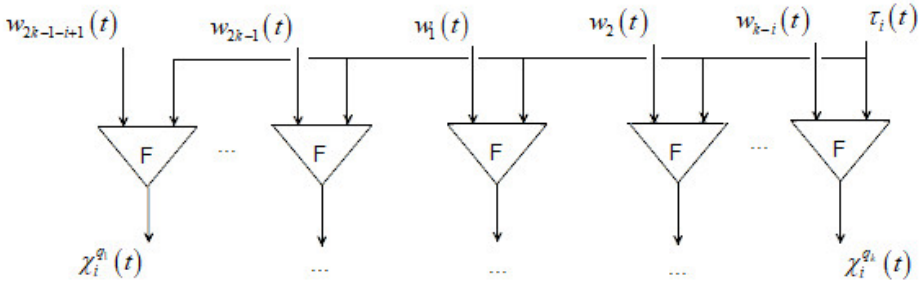


Рис. 10. Схема χ-многополюсника над скользящим окном

Рассмотрим принцип работы схемы на рис. 10. Так как элементов F в данной схеме ровно k штук и каждый элемент F дает ровно один выход в схеме, то получаем, что в схеме k штук выходов и каждый выход исходит из некоторого элемента F.

Далее, отметим, что $\tau_i(t)$ – это $i+1$-ая компонента вектора $\overline{y}_T(t)$, являющегося выходом блока $\mathbf{T}$. По определению, $\tau_i(t)=1 \iff t=k\cdot s+i$, где $i\in\{0,1,...,k-1\}$ подразумевается фиксированным числом, для которого строится рассматриваемый χ-многополюсник. Таким образом,

за одну эпоху только один раз на схему, изображенную на рис. 10, придет значение $\tau_i(t) = 1$.

Учитывая, что элемент F реализует функцию $\phi(x, y)$, обладающую теми свойствами, что $\phi(x, 1) = x$ и $\phi(x, 0) = 0$, то получаем, что верно (6).

$$\forall j \in \{1, ..., k\},\ \forall i' \in \{0, ..., i-1, i+1, ..., k-1\}\ \chi_i^{q_j}(k \cdot s + i') = 0 \qquad (6)$$

При этом, учитывая, что при $t = k{\cdot}s{+}i$ верно (5), а также определение F, получаем, что при $t = k{\cdot}s{+}i$ верно, что $\chi_i^{q_j} = q_j(k{\cdot}s)$ для $\forall j \in \{1, ..., k\}$. А из (6) следует, что в противном случае (при $t = k \cdot s + i',\ i' \neq i$) все выходы $\chi_i^{q_j}(t)$ блока χ-многополюсника с номером i равны нулю.

Таким образом, χ-многополюсники являются носителями векторов правильных состояний $\overline{q}(k{\cdot}s)$ в различные моменты времени одной эпохи.

В дальнейшем будем обозначать χ-многополюсники так, как это сделано на рис. 11.

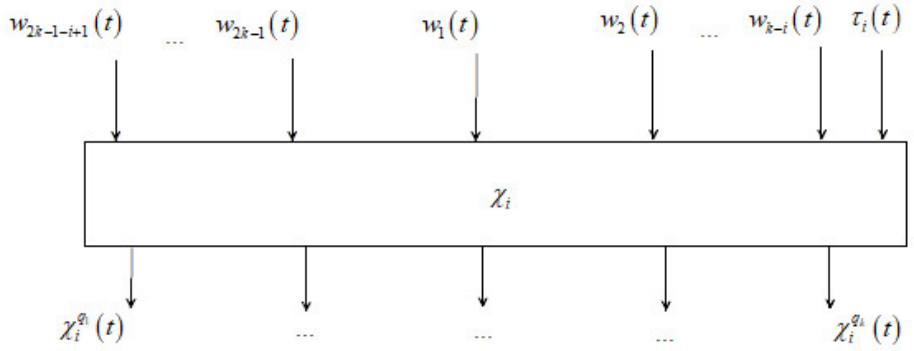


Рис. 11. Обозначение схемы χ-многополюсника

Далее построим χ-многополюсники для всех $i \in \{0, ..., k-1\}$ и направим их выходы, соответствующие одним и тем же $q_j,\ j \in \{1, ..., k\}$, в сумматор. Описываемая конструкция изображена на рис. 12.

Очевидно, что входы данной схемы соответствуют входам данного блока на рис. 2. Далее, покажем, что на выходах данного блока при любом $t = k \cdot s + i$ будет появляться сигнал $q_1(k \cdot s), ..., q_k(k \cdot s)$.

Так как у нас есть χ-многополюсники, соответствующие всем $i \in \{0, ..., k-1\}$, то при любом такте $t = k \cdot s + i$ на выходах хотя бы одного из них будут появляться значения $q_1(k \cdot s), ..., q_k(k \cdot s)$. Причем также по

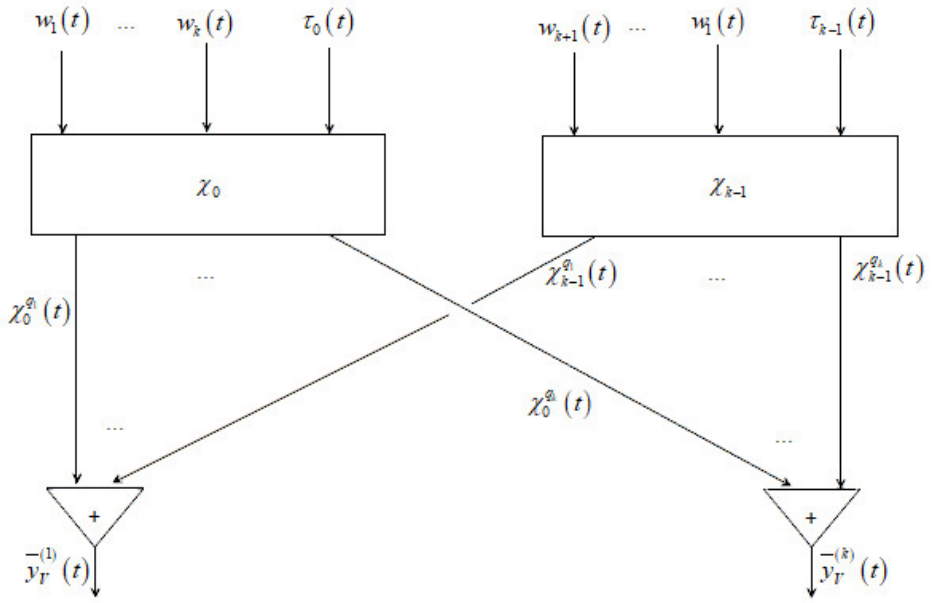


Рис. 12. Схема блока **V**

построению, все остальные χ-многополюсники в данный фиксированный момент времени будут давать на всех своих выходах нули.

Но тогда, так как, по построению схемы **V**, для любого $j \in \{1, ..., k\}$ верно, что $\overline{y}_V^{(j)}(t) = \sum\limits_{i' \in \{0,...,k-1\}} \chi_{i'}^{q_j}(t)$, то, в силу вышесказанного, $\overline{y}_V^{(j)}(k \cdot s + i) = \sum\limits_{i' \in \{0,...,k-1\}} \chi_{i'}^{q_j}(t) = \chi_i^{q_j}(k \cdot s + i) = q_j(k \cdot s)$. Откуда и следует доказываемое.

Отметим также, что при реализации данного блока не использовались задержки.

• Блок ψ. На блоке выпрямления **V** в начале каждой эпохи появляется корректное состояние канонической схемы. Однако во все остальные моменты времени, то есть при $t = k \cdot s + i,\ i \in \{1, ..., k-1\}$, получается, что в описанной на данный момент части схемы нет корректного состояния $\overline{q}(k \cdot s + i)$, поэтому вычислить $y(t)$, как значение $\psi(\overline{q}(t), \overline{x}(t))$ в произвольный момент времени мы не можем (так можно делать только в тактах, совпадающих с началом эпох). Блок ψ решает задачу вычисления $y(t)$ в любой момент времени $t = k \cdot s + i,\ i \in \{0, ..., k-1\}$.

Блок ψ строится на том свойстве, что по состоянию $\overline{q}(t)$ и последовательности входов $\overline{x}(t), \overline{x}(t+1), ..., \overline{x}(t+T)$ можно вычислить значение $y(t+T)$. Данные вычисления производятся функциями $\psi^{T+1}(\overline{q}(t), \overline{x}(t), ..., \overline{x}(t+T))$, канонические уравнения которых [1] записаны в выражении (7). В выражении (7) функции φ^{T-1} определяются аналогично функции φ^k, уравнения которой были записаны в выражении (4).

$$\begin{cases} \psi^T(\overline{q}(t-(T-1)), \overbrace{\overline{x}(t-(T-1)), ..., \overline{x}(t)}^{T}) = \\ \quad = \psi(\varphi^{T-1}(\overline{q}(t-(T-1)), \overline{x}(t-(T-1)), ..., \overline{x}(t-1)), \overline{x}(t)) \\ \psi^1(\overline{q}(t), \overline{x}(t)) = \psi(\overline{q}(t), \overline{x}(t)) \end{cases} \tag{7}$$

Переписывая выражение (7), взяв $t = k \cdot s + i$ и $T = i + 1$, получаем, что для вычисления $y(t) = y(k \cdot s + i)$, блок ψ должен принимать на вход значения $\overline{q}(k \cdot s)$ и набор значений $\overline{x}(k \cdot s), ..., \overline{x}(k \cdot s + i)$. Но значения $\overline{x}(k \cdot s), ..., \overline{x}(k \cdot s + i)$ выдаются в выходах блока **E**, а значения $\overline{q}(k \cdot s)$ выдаются блоком **V**.

Таким образом, имеются все необходимые входы для вычисления блока ψ в каждый момент времени, поэтому остается схемно реализовать все функции $\psi^i(\overline{q}(t-i+1), \overline{x}(t-i+1), ..., \overline{x}(t)),\ i \in \{1, ..., k\}$, а также реализовать правильный механизм выбора одного из данных значений в каждый момент времени внутри каждой эпохи.

Отметим сначала, что, так как $\varphi^i \in \Omega_0$ и $\psi \in \Omega_0$, то $\psi^i(\overline{q}(t-i+1), \overline{x}(t-i+1), ..., \overline{x}(t)) \in \Omega_0,\ i \in \{1, ..., k\}$, то есть для них существуют схемы из $K_0(B)$.

Теперь рассмотрим схему, изображенную на рис. 13.

Покажем, что схема, изображенная на рис. 13, на выходе всегда будет давать корректное значение $y(t)$, совпадающее с выходом канонической схемы в данный момент времени. Представим момент времени t в виде $t = k \cdot s + i$. Отметим, что, каждая функция $\psi^{i+1},\ i = 0, ..., k-1$ в данной схеме выдает нам значение $\psi^{i+1}(\overline{q}(k \cdot s), \overline{x}(k \cdot s), ..., \overline{x}(k \cdot s + i)) = y(k \cdot s + i)$ в момент времени $k \cdot s + i$.

При этом, так как все функции $\psi^{i+1},\ i = 0, ..., k-1$ есть в схеме, то в каждый момент времени $k \cdot s + i$ одна из функций $\psi^1, ..., \psi^k$ выдает корректное значение $y(k \cdot s + i)$.

Отметим, что выход каждой функции ψ^{i+1} подсоединен к первому входу элемента F, второй вход которого подсоединен к значению $\tau_i(t) \in \{0, 1\}$, причем $\tau_i(t) = 1 \iff t = k \cdot s + i$. Отсюда следует, что на выходе элемента F, подсоединенного к функции ψ^{i+1}, возникает значение $y(k \cdot s + i)$ в момент времени $t = k \cdot s + i$ и 0 в противном случае.

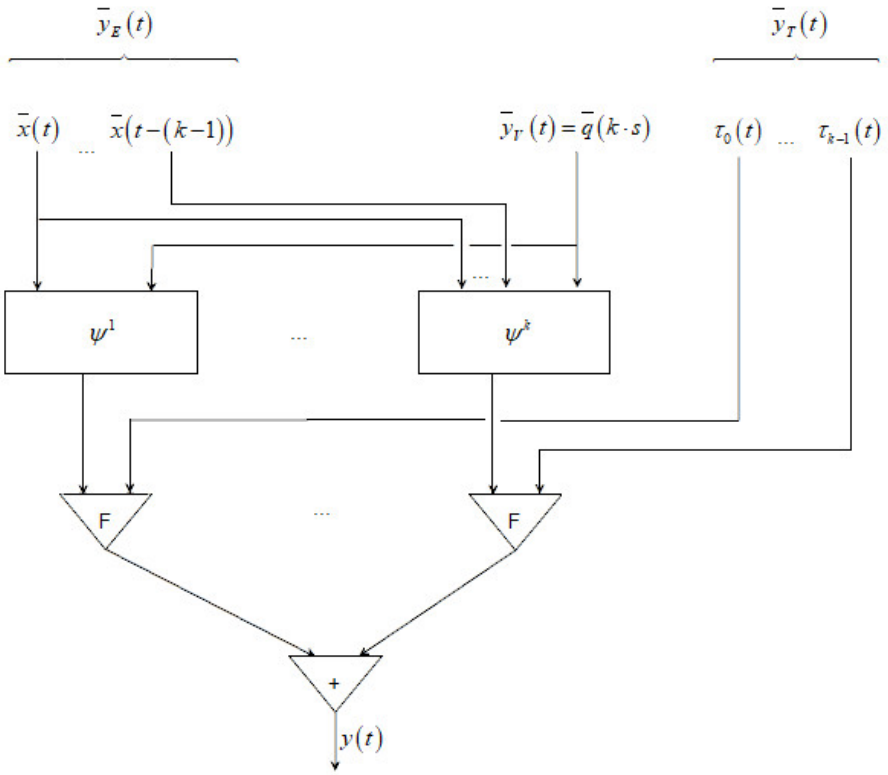


Рис. 13. Реализация блока ψ

Учитывая, что каждая функция ψ^{i+1}, $i = 0, ..., k-1$ подсоединена к своему элементу F и других элементов F в схеме нет, то получаем, что в каждый момент времени $t = k \cdot s + i$ выход некоторого элемента F равен $y(k \cdot s + i)$, а выходы всех остальных элементов вида F равны нулю.

Обозначая $y_i(t) = F(\psi^{i+1}(\overline{y}_V(t), \overline{y}_E^{(1),...,(i+1)}(t)), \tau_i(t))$ и учитывая вышесказанное, получаем, что на выходе схемы на рис. 13 в каждый момент времени $t = k \cdot s + i$ будет появляться значение $\sum\limits_{i'=1}^{k} y_{i'}(t) = y_i(t) + \sum\limits_{\substack{i'=1,\\ i' \neq i}}^{k} y_{i'}(t) = y(k \cdot s + i) + 0 = y(k \cdot s + i)$, откуда и следует доказываемое.

Отметим также, что при реализации данного блока не использовались задержки.

Таким образом, показано, что, если каждый из блоков схемы будет выдавать такие выходы, как на рис. 2, то схема будет выдавать корректное значение $y(t)$ в любой момент времени $t = 0, 1, 2, ...$ Откуда и следует доказываемое.

Причем отметим, у полученной схемы S_f^2 всего две обратные связи. Посчитаем теперь сложность полученной схемы.

Очевидно, что $L_2(f) \leqslant L(S_f^2) = L(\mathbf{T}) + L(\mathbf{E}) + L(\mathbf{S}) + L(\mathbf{R}) + L(\mathbf{V}) + L(\varphi^k) + L(\psi)$. Но тогда, учитывая, что $L(\mathbf{T}) = k, L(\mathbf{E}) = n \cdot (k-1), L(\mathbf{S}) = k, L(\mathbf{R}) = k-1, L(\mathbf{V}) = 0, L(\varphi^k) = 0, L(\psi) = 0$, получаем оценку $L_2(f) \leqslant k + n \cdot (k-1) + k + (k-1) = (n+3) \cdot k - n - 1 \leqslant (n+3) \cdot k = (n+3) \cdot L(f)$.

Таким образом, нашлась константа $c = n + 3$, что $L_2(f) \leqslant L(f)$. □

Таким образом, установлено, что при фиксированном количестве входов функций из Ω количество задержек в минимальной (по числу задержек) для этой функции схеме, с не более чем двумя обратными связями, растет линейно от количества задержек в заданной схеме, реализующей ту же функцию.

Теорема 4. *Любой конечный автомат с n входами, реализуемый схемой, содержащей k задержек, в базисе из задержек и полного множества функциональных элементов, можно задать схемой с двумя обратными связями и $n \cdot (k-1) + 2k - 1 + \lceil \log_2\ k \rceil$ задержками.*

Любую нейронную схему с памятью, реализуемую из линейных функциональных элементов, вентилей и k задержек, можно задать схемой с одной обратной связью в соответствующем базисе, содержащей $n \cdot k + 2k + 1$ задержек.

Доказательство. Оценка $n \cdot (k-1) + 2k - 1 + \lceil \log_2\ k \rceil$ вытекает из теоремы 3. Действительно, в предъявленной на рис. 2 схеме выполняется $L(\mathbf{T}) = k, L(\mathbf{E}) = n \cdot (k-1), L(\mathbf{S}) = k, L(\mathbf{R}) = k-1, L(\mathbf{V}) = 0, L(\varphi^k) = 0, L(\psi) = 0$. При этом, очевидно, что блок $\mathbf{T}$ можно реализовать со сложностью $\lceil \log_2\ k \rceil$ (см. замечание в доказательстве теоремы 3).

Тогда схема S_f^2 с двумя обратными связями имеет сложность $L(S_f^2) = \lceil \log_2\ k \rceil + n \cdot (k-1) + k + (k-1) = n \cdot (k-1) + 2k - 1 + \lceil \log_2\ k \rceil$.

Вторая оценка $n \cdot k + 2k + 1$, верная для нейронных схем с памятью, получается из того факта, что в случае нейронных схем с памятью блок $\mathbf{T}$ можно реализовать без использования обратных связей [2]. Однако взамен такой реализации требуется изменить каноническую схему, добавив в нее еще одну задержку. Таким образом, получается, что схема с одной обратной связью строится с использованием канонической схемы с $k' = k + 1$ задержкой. Ее сложность равна $L(S_f^2) = n \cdot (k'-1) + 2k' - 1 = n \cdot k + 2k + 1$. □

Список литературы

[1] Кудрявцев В.Б., Алешин С.В., Подколзин А.С., *Введение в теорию автоматов*, «Наука», Москва, 1985, 320 с.

[2] Половников В.С., *Об оптимизации структурной реализации нейронных сетей*, Диссертация на соискание степени кандидата наук, Москва, 2007, 96 с.

[3] Саймон Хайкин, *Нейронные сети. Полный курс*, «Вильямс», Москва, 2006, 1104 с.

Feedback connection in recurrent circuits

Aleksiadis N.P., Polovnikov V.S., Chasovskikh A.A., Shishlyakov V.G.

It is shown that, under some natural restrictions on the element basis, for any recurrent circuit with a fixed number of inputs, it is possible to construct a circuit functionally equivalent to it in the same basis using superposition operations and no more than a double application of the feedback operation with a linear increase in the number of delays used compared to original circuit implementation. Thus, the linearity of the order of memory growth in the transition to the optimal (in terms of the number of delays) scheme with no more than two feedbacks has been proved. Short-term and long-term memory modules are distinguished in the structure of the recurrent circuits with no more than two feedbacks constructed in the work. The result obtained, in particular, is valid for the class of finite automata, as well as for the class of neural circuits built from elements containing gates.

Keywords: recurrent circuits, feedback, linear estimation, gates.

References

[1] Kudryavtsev V.B., Alyoshin S.V., Podkolzin A.S., *Introduction to automata theory*, «Science», Moscow, 1985 (In Russian), 320 pp.

[2] Polovnikov V.S., *On optimization of the structural implementation of neural networks*, Ph.D. Thesis ... physical and mathematical sciences, MSU, Moscow, 2007 (In Russian)

[3] Simon Haykin, *Neural Networks. A Comprehensive Foundation*, «Williams», Moscow, 2006 (In Russian), 1104 pp.

Распределенный поиск компонент сильной связности в адаптивной модели

И. В. Молдованов[1]

В работе исследуется сложностная оценка алгоритма поиска компонент сильной связности в Adaptive Massively Parallel Computations(AMPC) модели. Данная модель, в отличие от иных более ограниченных распределенных формализаций, позволяет в рамках одного шага алгоритма строить дерево запросов в распределенную память. Получен вероятностный алгоритм, реализующий поиск компонент сильной связности за полилогарифмическое или сублинейное время, в зависимости от объема доступной локальной памяти. Объем требуемой локальной памяти является сублинейной величиной по отношению к числу вершин в графе.

Ключевые слова: распределенные алгоритмы, вероятностные алгоритмы, компоненты сильной связности.

1. Введение

В современном мире вычислительные мощности растут значительно быстрее чем доступные ресурсы памяти, что в существенной мере ограничивает возможность обрабатывать большие объемы данных. В данном ключе особый интерес получают распределенные системы, позволяющие более эффективным способом использовать значительное число вычислительных единиц (ядер, процессоров и т.д.). Чтобы иметь возможность разрабатывать более эффективные алгоритмы для подобных систем, был предложен ряд математических формализаций: CONGEST, CONGESTED-CLIQUE, MPC (Massively Parallel Computations) и, наконец, AMPC (Adaptive Massively Parallel Computations). Последняя модель была предложена сравнительно недавно в статье [3], и обладает наибольшими выразительными возможностями, позволяя эмулировать все предыдущие формализации. В рамках данной работы будет представлен алгоритм поиска компонент сильной связности направленного графа, выраженный в AMPC модели, и представлена оценка его сложности.

[1] *Молдованов Илья Владимирович* — студент каф. математической теории интеллектуальных систем мех.-мат. ф-та МГУ, e-mail: ilmoldovanov@mail.ru.

Moldovanov Ilya Vladimirovich — student, Lomonosov Moscow State University, Faculty of Mechanics and Mathematics, Chair of Mathematical Theory of Intellectual Systems.

1.1. Распределенные вычисления

Ранее уже была обозначена причина бурного развития интереса к распределенным вычислениям. Остановимся теперь более подробно на описанных выше формализациях. Первой значимой вехой в развитии данных систем была CONGEST модель. Ее первичное предназначение - описание графовых алгоритмов, что, в свою очередь, привело к следующей структуре: каждая вершина исходного графа представляет собой отдельный вычислительный узел, обладающий бесконечной памятью. Коммуникация между узлами осуществляется через ребра исходного графа, и ее размер ограничен $O(\log n)$ битами, где n - число вершин в графе. В качестве меры сложности в данной модели принимается число раундов подобных коммуникаций. Несмотря на кажущееся неудобство подобного подхода с точки зрения практики, данная формализация позволяет достаточно хорошо формулировать сложностные оценки для алгоритмов в терминах Pregel описания[1], которое оказало значительное влияние на сферу проектирования распределенных алгоритмов на графах. В свою очередь, более общая CONGESTED-CLIQUE модель сохраняет все ограничения CONGEST формализации, однако позволяет проводить коммуникация между всеми парами вершин, независимо от наличия соответствующих ребер в исходном графе.

Следующим этапом развития распределенных алгоритмов можно считать появление MPC модели, впервые описанной в [2]. Данная модель подразумевает использование вычислительных узлов с распределенной памятью уже в более каноничном, по сравнению с предыдущими формализациями, ключе и обычно характеризуется тремя параметрами: P - количество доступных машин, N - размер входных данных и S - объем памяти на каждой машине, а так же максимально число отправляемых сообщений (более точно: каждая машина может хранить в памяти и отправить за один раунд не более $O(S \log n)$ бит). Наибольший интерес, очевидно, представляют значения параметра $S << N$, таким образом в большинстве работ рассматривается строго сублинейный случай, то есть $S = N^{\epsilon}$, для некоторого $0 < \epsilon < 1$. В качестве основной меры сложности для оценки MPC алгоритмов принимается количество раундов, в рамках которых происходит обмен сообщениями между машинами и произвольные локальные вычисления.

Описанные в данном разделе модели, несмотря на значимые структурные отличия, обладают ключевым общим свойством - сложность алгоритмов в данных формализациях определяется объемом коммуникаций между вычислительными машинами, а не количеством арифметических операций или обращений к памяти. Данная особенность в вышеизложенном описании формализаций отражена в том факте, что, вообще говоря,

локальные вычисления в рамках раунда допускаются в произвольном объеме. Подобный подход к моделированию распределенных вычислений объясняется спецификой соответствующих программных реализаций - при наличии большого количества вычислительных узлов(каждый из которых может обладать несколькими потоками) именно коммуникация между данными узлами становится основным узким местом большинства задач. Тем не менее чтобы сохранить связь с приложениями, в литературе, посвященной распределенным алгоритмам в той или иной формализации, хорошим тоном считается отмечать сложность локальных вычислений, если она может быть высоко-полиномиальной или даже экспоненциальной.

Замечание В данной работе сложность локальных вычислений не превосходит сублинейной по отношению к числу вершин в графе.

Таким образом, сложность распределенных алгоритмов не является понятием тождественным сложности алгоритмов в ее классическом понимании, однако при экстремальных значениях параметров модели (например, если система состоит только из одной машины, которой разрешено провести только одну операцию в раунд) данные понятия можно считать достаточно близкими (особенно это касается MPC и AMPC моделей). Дополнительно подобную аналогию можно провести так же и с параллельными парадигмами вычислений, например PRAM, так в работе [2] приводятся симуляции некоторых PRAM алгоритмов в MPC формализации.

Таким образом, мы косвенным путем получаем верхнюю оценку сложности распределенных алгоритмов, по крайней мере для MPC и AMPC моделей, которые являются предметом фокуса данной работы.

1.2. AMPC модель

В данной работе мы будем рассматриват AMPC модель, отличительные особенности которой представлены ниже. Впервые описанная в [3], данная формализация обобщает MPC подход, позволяя без увеличения сложности в терминах раундов эмулировать любые реализуемые в MPC алгоритмы.

Данная модель характеризуется следующими параметрами: N - размер входных данных, P - количество машин, которые будут обрабатывать данные, S - объем памяти на каждой машине выражаемый, в количестве идентификаторов, размера $\log N$ бит, которые могут храниться в памяти машины. Обозначим также через T общий объем памяти всех машин, то есть $T = SP$.

В данной работе мы рассмотрим наиболее интересный случай $S = O(N^{\epsilon})$, $0 < \epsilon < 1$. Подобное соотношение параметров характеризует ситуацию,

когда все данные не могут быть целиком размещены в памяти одной машины, то есть начинают оказывать влияние естественные для распределенных алгоритмов ограничения, связанные с необходимостью синхронизации результатов вычислений, произведенных каждой машиной, а так же отсутствием доступа ко всем данным. Более того, в рамках данной работы будет представлен алгоритм, позволяющий решать исходную задачу в ситуации, когда $S = O(n^\epsilon)$, $0 < \epsilon < 1$, где n - число вершин исследуемого графа.

Также в рамках AMPC модели имеется набор распределенных неограниченных по памяти хранилищ данных, которые мы обозначим через $D_0, D_1, D_2, ...$ Для описания удобно предположить, что все хранилища предоставляют семантику доступа ключ-значение, то есть, что они хранят набор пар ключ-значение и по запросу, содержащему заданный ключ, возвращают соответствующий идентификатор. При этом важно отметить, что размеры ключа и индентификатора составляют $O(\log N)$ бит. В свою очередь входные данные хранятся в D_0 и используют набор ключей известный всем машинам (например, последовательные целые числа). Дополнительно при наличии $k > 1$ пар ключ-значение, имеющих один и тот же ключ x, отдельные значения могут быть получены, по запросам $(x, 1), ..., (x, k)$. Стоит отметить, что индексы от 1 до k присваиваются произвольно. Наконец, запрос ключа, которого нет в распределенном хранилище, приводит к пустому ответу.

Алгоритм в данной модели состоит из раундов. В i-ом раунде каждая машина может считать данные из D_{i-1} записать в D_i. Таким образом, в течение раунда каждая машина может выполнить до $O(S)$ операций чтения (далее называемых запросами), произвести $O(S)$ записей, а так же выполнить произвольные вычисления. Каждый запрос и запись представляют собой обращение к хранилищу для чтения или записи одной пары ключ-значение. Ключевым свойством модели по сравнению с MPC формализацией является то, что запросы, которые машина делает в течение раунда, могут зависеть от результатов предыдущих запросов, сделанных в том же раунде.

1.3. Пример алгоритма в AMPC модели

Приведем теперь наглядный пример функционирования описанной выше модели.

Рассмотрим граф G, имеющий n вершин и m ребер. Задача состоит в том, чтобы вычислить степень каждой вершины в данном графе.

Зафиксируем параметры модели: размер задачи $N = n + m$, число машин $P = n$ и объем локальной памяти $S = n^\epsilon$, для некоторого $0 < \epsilon < 1$. В качестве начального состояния памяти D_0 выберем следующее пред-

ставление графа: граф G записан в виде набора ребер, то есть пар вида $(Source, Destination)$. В данном случае ключ $Source$, и соответствующая запись $Destination$ представляют собой целые числа, не превышающие n, а значит удовлетворяют ограничению на длину записи $O(\log N)$
Отметим, что представление в виде списка смежности было бы некорректно в данной модели, так как размер записи для каждого ключа составлял бы $O(n \log N)$ бит.
Рассмотрим теперь возможный алгоритм решения поставленной задачи. Пронумеруем каждую машину от 1 до n и установим соответствие между машинами и вершинами с совпадающими номерами. Для подсчета степеней вершин каждой машине необходимо считать все ребра с соответствующим значением ключа, увеличивая внутренний счетчик. Таких ребер может быть не более чем $n - 1$, а значит за $O(n^{1-\epsilon})$ раундов каждая машина сможет прочесть все ассоциированные с данной вершиной ребра. В качестве ответа каждая машина запишет пару $(Source, Degree)$ в конечное состояние общей памяти.

1.4. Мотивация применения AMPC модели

Дополняя представленный выше пример, опишем более подробно применимость AMPC модели для решения практических задач.
В мире распределенных вычислений значительную роль играет понятие "MapReduce". Данный фреймворк позволяет реализовать параллельные вычисления над большими объемами данных с использованием набора вычислительных узлов, формирующих кластер. Подобные кластера могут содержать значительное количество компьютеров, связанных беспроводным соединением. Тем не менее память каждой отдельной машины ограничена размером оперативной памяти или диска, что позволяет единовременно хранить лишь фрагмент общих данных. Таким образом, описанная ранее MPC модель была впервые представлена как формализация подобного подхода к распределенным вычислениям, позволяющая оценить эффективность различных алгоритмических решений.
В свою очередь, AMPC модель была представлена как развитие основополагающих идей предыдущей формализации. Данная модель была дополнена распределенной базой данных, позволяющей в некотором роде устранить недостаток, связанный с ограничением локальной памяти на машинах, за счет относительно быстрых запросов в распределенное хранилище.

1.5. Постановка задачи

Подводя итог представленному выше описанию AMPC модели, обозначим описанные в работах [3] и [4] результаты.

Авторам данных исследований удалось в AMPC модели получить эффективные реализации большого количества фундаментальных задач, присущих ненаправленным графам. В числе описанных ими алгоритмов присутствуют Связность, Связность леса, Нахождение минимального покрывающего дерева, Нахождение максимального независимого множества, Поиск связных компонент. Важно отметить, что сложность полученных реализаций превосходит(в плане эффективности) ранее известную сложность в MPC формализации.
Таким образом, в качестве потенциального направления для дальнейшего исследования можно выделить рассмотрение аналогичных задач в направленных графах. В данном ключе естественным кандидатом является фундаментальная задача поиска компонент сильной связности, которой уделяется значительное внимание в классической литературе. Еще более интересным данное направление исследования делает тот факт, что на данный момент не удалось получить эффективную MPC реализацию подобного алгоритма(по сложности превосходящую параллельные аналоги). Таким образом, целью данной работы была поставлена следующая задача: используя превосходящие выразительные возможности AMPC модели, получить распределенный алгоритм поиска компонент сильной связности, имеющий, как максимум, сублинейную сложность.

1.6. Компоненты сильной связности

Сначала напомним ряд определений.
Ориентированный граф $G(V, E)$ называется **сильно связным**, если для любых двух вершин $s, t \in V$ существуют направленные пути из s в t и из t в s.
Компонентами сильной связности ориентированного графа называются его максимальные по включению сильно связные подграфы.
Простейший алгоритм для решения данной задачи состоит в следующем: для каждой пары вершин в графе необходимо определить взаимную достижимость и соединить пары, удовлетворяющие данному свойству, ненаправленным ребром. Далее алгоритм поиска компонент связности на построенном ненаправленном графе позволяет найти искомый ответ.
Среди более совершенных подходов можно выделить линейные(в сложностных терминах классической последовательной парадигмы) алгоритмы Тарьяна [5] и Косарайю [6].
Тем не менее, несмотря на свою вычислительную эффективность, данные алгоритмы плохо подходят для параллельной реализации, так как являются последовательными по своей сути. Таким образом, первый параллельный алгоритм поиска компонент сильной связности был предложен в [7]. В качестве основного узкого места данного подхода можно вы-

делить линейную по числу вершин глубину рекурсии, в худшем случае. Тем не менее данная проблема была успешно решена в [8]. Авторам удалось представить алгоритм, позволяющий реализовать поиск компонент сильной связности в CRCW PRAM модели со сложностью $O(R \log^2 n)$, где R - сложность алгоритма нахождения всех вершин в графе, достижимых из заданной.

1.7. Структура работы

В заключение вводной части приведем план дальнейшего изложения результатов, представленных в данной работе.
Определение сложности полученного в работе алгоритма, а так же доказательство его корректности во многом базируются на сформулированных и доказанных в разделе **2 Выборка вершин** технических утверждениях.
Основная содержательная часть работы изложена в разделе **3 Достижимость из одной вершины**. В рамках данного раздела приведен распределенный алгоритм, позволяющий определить множество достижимости для фиксированной вершины в графе. В подразделе **3.2** представлена версия алгоритма, позволяющая решить поставленную задачу только для графа с фиксированными показателями входящих и исходящих степеней, однако в подразделах **3.3** и **3.4** описаны способы обобщить предыдущий результат на произвольный граф.
Наконец, заключительный результат представлен в виде описанного в разделе **4 Компоненты сильной связности** подхода к поиску компонент сильной связности в AMPC модели.

2. Выборка вершин

В данной секции мы рассмотрим вероятностный подход к выборке вершин в графе, который позволяет, практически не увеличивая общую сложность алгоритма, сконструировать подмножество вершин, обладающих необходимыми нам полезными свойствами. Данная техника эффективно применяется в ряде вероятностных графовых алгоритмов, при этом особенный интерес в контексте поставленной задачи для нас представляют работы [9] и [10].
Приведем сперва два ключевых результата, которые легли в основу всех утверждений данной секции:

Теорема 1. ***Неравенство Чернова*** *Пусть* $X = \sum_{i=1}^{k} X_i$, *где все* X_i *независимые Бернуллиевские случайные величины, принимающие значение*

1 с вероятностью p_i, и значение 0 с вероятностью $1 - p_i$. Через μ обозначим $E[X] = \sum_{i=1}^{k} E[X_i]$, тогда:

1) $P(X \geq (1+\delta)\mu) \leq e^{-\frac{\delta^2\mu}{2+\delta}}$ *для всех* $\delta > 0$

2) $P(X \leq (1-\delta)\mu) \leq e^{-\frac{\delta^2\mu}{2}}$ *для всех* $0 < \delta < 1$

Следствие 1. *Пусть X, X_i и μ соответствуют условиям теоремы, тогда:*

$$P(|X - \mu| \geq \delta\mu) \leq 2e^{-\frac{\delta^2\mu}{3}} \text{ для всех } 0 < \delta < 1$$

Более подробно данные утверждения изложены в [11].

Лемма 1. *Рассмотрим V - множество натуральных чисел от 1 до n включительно и константы $\gamma \leq n, \delta \in (0,1)$. Пусть так же имеется L - проиндексированный набор из n подмножеств V, таких что размер каждого не меньше чем γ, то есть $\forall L_v \in L$ имеем $L_v \subseteq V, |L_v| \geq \gamma$. Далее рассмотрим набор независимых одинаково распределенных случайных величин S_v, проиндексированных $1...n$ и имеющих распределение Бернулли с параметрами*

$$S_v = \begin{cases} 1, & p \\ 0, & 1-p \end{cases}$$

Тогда если $p \geq \min(\frac{12\log(2n/\delta)}{\gamma}, 1)$, то с вероятностью как минимум $1-\delta$

1) $\forall L_v \in L\ \exists u \in L_v :\ S_u = 1.$

2) $\sum_{v \in V} S_v \in [\frac{1}{2}pn, \frac{3}{2}pn]$

Доказательство. 1) Для фиксированного индекса v рассмотрим $Pr(\sum_{u \in L_v} E[S_u] - \sum_{u \in L_v} S_u \geq \frac{1}{2}\sum_{u \in L_v} E[S(u)])$ Поскольку S_v - независимые, одинаково распределенные Бернуллиевские случайные величины, то воспользовавшись оценкой Чернова получим следующее неравенство:

$$Pr(|\sum_{u \in L_v} E[S_u] - \sum_{u \in L_v} S_u| \geq \frac{1}{2}\sum_{u \in L_v} E[S_u]) \leq \tag{1}$$

$$2\exp\left(-\frac{\sum_{u \in L_v} E[S_u]}{12}\right) = 2\exp\left(-\frac{p|L_v|}{12}\right) \leq \tag{2}$$

$$2\exp\left(-\frac{p\gamma}{12}\right) \leq 2\exp\left(-\log 2n/\delta\right) \leq \frac{\delta}{n} \tag{3}$$

Поскольку $\frac{1}{2}\sum\limits_{u\in L_v} E[S(u)] \geq 1$, то получаем, что с вероятностью как минимум $1-\frac{\delta}{n}$, $\sum\limits_{u\in L_v} S_u \geq 1$.

Наконец, объединив вероятности для всех $L_v \in L$, пункт (1) выполняется с вероятностью как минимум $1-\delta$

2) Рассмотрим выражение $Pr(|\sum\limits_{v\in V} E[S_v] - \sum\limits_{v\in V} S_v| \geq \frac{1}{2}\sum\limits_{v\in V} E[S(v)])$ Поскольку S_v - независимые, одинаково распределенные Бернуллиевские случайные величины, то воспользовавшись оценкой Чернова получим следующее неравенство:

$$Pr(|\sum_{v\in V} E[S_v] - \sum_{v\in V} S_v| \geq \frac{1}{2}\sum_{v\in V} E[S_v]) \leq \tag{4}$$

$$2\exp\left(-\frac{\sum\limits_{v\in V} E[S_v]}{12}\right) = 2\exp\left(-\frac{p|V|}{12}\right) \leq \tag{5}$$

$$2\exp\left(-\frac{p\gamma}{12}\right) \leq 2\exp\left(-\log 2n/\delta\right) \leq \frac{\delta}{n} \leq \delta \tag{6}$$

□

Рассмотрим направленный граф $G = G(V, E)$. Для каждой вершины $v \in V$ обозначим $D_{out}(v)$ - множество ее соседей по исходящим ребрам, а $D_{in}(v)$ - по входящим, в дополнение введем $Deg_{out}(v) = |D_{out}(v)|$ и $Deg_{in}(v) = |D_{in}(v)|$.
В свою очередь исходящей степенью графа $Deg_{out}(G)$ назовем $\min\limits_{v\in V} Deg_{out}(v)$. Аналогично определим $Deg_{in}(G)$.

Лемма 2. *Рассмотрим направленный граф $G = G(V, E), |V| = n$, и параметры $d \in [\log n, n-1]$, $\delta \in (0, 1)$. Если $Deg_{out}(G) \geq d$, то с вероятностью как минимум $1-\delta$ $\exists SV \subset V$, такое что $\forall v \in V \exists u \in D_{out}(v) \cup \{v\}$: $u \in SV$ и $|SV| \in O(\frac{n\log n}{d})$*

Доказательство. Для доказательства рассмотрим **Утверждение 1**. Множество натуральных чисел V представляет собой номера вершин в графе, а в качестве семейства подмножеств L выберем номера соседей, составляющих исходящую окрестность каждой вершины. Затем, в качестве p_d возьмем число $\frac{c\log n}{d}$, где c - наименьшее такое целое число, что $\frac{c\log n}{d} \geq \frac{12\log(2n/\delta)}{d}$. Наконец во множество SV добавим только такие вершины, индексы которых соответствуют Бернуллиевским случайным величинам S_v, принявшим значение "1"(вероятность этого события оставляет p_d). Тогда:

- По пункту 1 **Утверждения 1** получим, что каждое множество из L с вероятностью как минимум $1-\delta$ содержит индекс, который соответствует Бернуллиевской случайно величине, принявшей значение "1". Таким образом, каждая исходящая окрестность в графе с вероятностью как минимум $1-\delta$ содержит вершину из SV.

- По пункту 2 **Утверждения 1** $|SV| \in O(p_d d)$ с вероятностью как минимум $1-\delta$, а так как $p_d = \frac{c \log n}{d} \in O(\frac{logn}{d})$, то $|SV| \in O(n\frac{\log n}{d})$ с вероятностью как минимум $1-\delta$

□

Теперь нам необходимо доказать более жесткое свойство. В данном случае мы хотим показать, что для некоторого d мы можем таким образом выбрать подмножество вершин, что если из заданной вершины v в любую другую вершину в графе существует хотя бы один направленный путь, превышающий по длине d, то так же из v в данную вершину с большой вероятностью существует и такой путь, что любой его подотрезок, длина которого превышает $2d$, содержит хотя бы одну вершину из случайно выбранного подмножества. На основании вышеизложенных рассуждений приведем следующее утверждение. Стоит отметить, что фактически, в плане доказательства, оно мало отличается от описанного ранее, так как обозначенное увеличение количества множеств повлияет лишь на константу в выражении для вероятности.

Лемма 3. *Пусть имеется V - набор индексов от $1...n$ и константы $\gamma \leq n, \delta \in (0,1)$. Пусть так же имеется L - проиндексированный набор из n^2 подмножеств V, таких что размер каждого не меньше чем γ, то есть $\forall L_{v,i} \in L$, $1 \leq v \leq n$, $1 \leq i \leq n$ имеем $L_{v,i} \subseteq V, |L_{v,i}| \geq \gamma$. Далее рассмотрим набор независимых одинаково распределенных случайных величин S_v проиндексированных $1...n$ и имеющих распределение Бернулли с параметрами*

$$S_v = \begin{cases} 1, & p \\ 0, & 1-p \end{cases}$$

Тогда если $p \geq \min(\frac{12 \log(2n^2/\delta)}{\gamma}, 1)$, то с вероятностью как минимум $1-\delta$

1) $\forall L_v \in L \; \exists u \in L_v : \; S_u = 1$.

2) $\sum\limits_{v \in V} S_v \in O(pn)$

Доказательство. 1) Для фиксированных индексов v, i рассмотрим $Pr(\sum\limits_{u\in L_{v,i}} E[S_u] - \sum\limits_{u\in L_{v,i}} S_u \geq \frac{1}{2}\sum\limits_{u\in L_{v,i}} E[S(u)])$ Поскольку S_v - независимые, одинаково распределенные Бернуллиевские случайные величины, то воспользовавшись оценкой Чернова получим следующее неравенство:

$$Pr(|\sum_{u\in L_{v,i}} E[S_u] - \sum_{u\in L_{v,i}} S_u| \geq \frac{1}{2}\sum_{u\in L_{v,i}} E[S_u]) \leq \tag{7}$$

$$2\exp\left(-\frac{\sum\limits_{u\in L_{v,i}} E[S_u]}{12}\right) = 2\exp\left(-\frac{p|L_{v,i}|}{12}\right) \leq \tag{8}$$

$$2\exp\left(-\frac{p\gamma}{12}\right) \leq 2\exp\left(-\log 2n^2/\delta\right) \leq \frac{\delta}{n^2} \tag{9}$$

Поскольку $\frac{1}{2}\sum\limits_{u\in L_{v,i}} E[S(u)]) \geq 1$, то получаем, что с вероятностью как минимум $1 - \frac{\delta}{n^2}$, $\sum\limits_{u\in L_{v,i}} S_u \geq 1$.

Наконец, объединив вероятности для всех $L_{v,i} \in L$, пункт (1) выполняется с вероятностью как минимум $1 - \delta$

2) Рассмотрим выражение $Pr(|\sum\limits_{v\in V} E[S_v] - \sum\limits_{v\in V} S_v| \geq \frac{1}{2}\sum\limits_{v\in V} E[S(v)])$ Поскольку S_v - независимые, одинаково распределенные Бернуллиевские случайные величины, то воспользовавшись оценкой Чернова получим следующее неравенство:

$$Pr(|\sum_{v\in V} E[S_v] - \sum_{v\in V} S_v| \geq \frac{1}{2}\sum_{v\in V} E[S_v]) \leq \tag{10}$$

$$2\exp\left(-\frac{\sum\limits_{v\in V} E[S_v]}{12}\right) = 2\exp\left(-\frac{p|L_v|}{12}\right) \leq \tag{11}$$

$$2\exp\left(-\frac{p\gamma}{12}\right) \leq 2\exp\left(-\log 2n^2/\delta\right) \leq \frac{\delta}{n^2} \leq \delta \tag{12}$$

□

Лемма 4. *Рассмотрим направленный граф $G = G(V, E), |V| = n$ и вершину $v \in V$, а так же параметры $d \in [\log(n), n], \delta \in (0, 1)$. Тогда $\exists SV \subset V$, такое, что если в графе существует ацикличный направленный путь $P(v, u) : |P| \geq d$, то с вероятностью как минимум*

$1-\delta$ *существует такой путь* $P'(v,u)$, $|P'| \geq d$, *что любой подучасток данного пути длины как минимум* $2d$ *содержит вершину из* SV *и* $|SV| \in O(\frac{n \log n}{d})$.

Доказательство. Для доказательства рассмотрим **Утверждение 3**. В качестве семейства подмножеств L выберем последовательные подотрезки длины d на путях из v во все остальные вершины. Затем, в качестве p_d возьмем число $\frac{c \log n}{d}$, где c - наименьшее такое целое число, что $\frac{c \log n}{d} \geq 12 \log(2n^2/\delta)$. Тогда:

- Как было сказано выше, множество L представляет собой последовательные подотрезки длины d на путях из v во все остальные вершины. Если же пути соответствующей длины из вершины v в некоторую другую вершину нет, то без ограничения общности заполним данные $n-1$ множество произвольными вершинами.
  Итак, $|L| \leq n^2$, а значит, применив пункт 1 **Утверждения 3**, получим, что для некоторого пути из v в любую другую вершину, его последовательные подотрезки длины d содержат как минимум одну вершину из SV. Таким образом, расстояние между двумя вершинами из SV на данном пути не превышает $2d$.

- По пункту 2 **Утверждения 3** $|SV| \in O(p_d n)$,а так как $p = \frac{c \log n}{d} \in O(\frac{\log n}{d})$, то $|SV| \in O(\frac{n \log n}{d})$

□

3. Достижимость из одной вершины

В данной секции мы опишем вероятностный алгоритм в AMPC модели, позволяющий для графа $G = G(V,E)$ найти подмножество вершин, достижимое из фиксированной вершины $v \in V$.

3.1. Обзор структуры алгоритма

Сперва мы идейно представим ход алгоритма поиска всех вершин, достижимых из фиксированной вершины v, а так же кратко опишем идеи, лежащие в его основе.
Будем считать далее, что каждая вершина в графе имеет фиксированные входящую и исходящую степень, равные $\lceil n^{\frac{\epsilon}{2}} \rceil$(в последующих секциях будет так же приведен обобщенный алгоритм, тем не менее сохраняющий все основные идеи и принципы описываемого подхода).
Опираясь на материал, изложенный в предыдущей секции, за $O(1)$

раундов в AMPC модели мы можем построить такое подмножество $SV(n^{1-\epsilon}\log n)$, что если между двумя вершинами в графе есть путь, по длине превышающий $2\lceil n^{1-\epsilon}\log n\rceil$, то расстояние на этом пути между последней вершиной из $SV(n^{1-\epsilon}\log n)$ и конечной вершиной с большой вероятностью не превышает $2\lceil n^{1-\epsilon}\log n\rceil$. Таким образом, задача разобьется на два этапа:

- для каждой вершины в графе найти все вершины из $SV(n^{1-\epsilon}\log n)$, находящиеся на расстоянии не превышающем $2\lceil n^{1-\epsilon}\log n\rceil$, из которых она достижима
- для каждой вершины из $SV(n^{1-\epsilon}\log n)$ сказать, достижима ли она из исходной вершины v

В таком случае, по транзитивности мы сможем для каждой вершины в графе с вероятностью как минимум $1-\delta$, где $\delta \in O(1)$, сказать, достижима ли она из вершины v.
Рассмотрим сперва решение первой подзадачи. По **Утверждению 4** размер множества $SV(n^{1-\epsilon}\log n)$ с большой вероятностью не превышает $O(n^{\epsilon})$. То есть при параллельном поиске в ширину сразу из всех вершин множества $SV(n^{1-\epsilon}\log n)$, все уникальные идентификаторы могут быть укомплектованы в память или запрос одной машины. Выберем теперь подмножество $SN(n^{\frac{\epsilon}{2}})$, такое, что для любой вершины в графе, либо она сама, либо ее входящий сосед будут принадлежать данному подмножеству с вероятностью как минимум $1-\delta$. По **Утверждению 2** данную операцию мы так же можем произвести за $O(1)$ раундов в AMPC модели. Вершины из $SN(n^{\frac{\epsilon}{2}})$ мы назовем опорными, и каждую вершину в графе мы свяжем с минимальной по номеру опорной вершиной в ее окрестности(мы уже гарантировали, что объект для установления связи найдется с большой вероятностью). Каждую опорную вершину и все связанные с ней простые вершины мы назовем гипервершиной, далее соединим направленными ребрами все такие гипервершины, составные вершины которых имели хотя бы одно направленное ребро. По **Утверждению 2** количество вершин в получившимся гиперграфе (механизм отождествления вершин и обхода гиперграфа мы опишем подробно в соответствующем разделе, однако отметим, что данная процедура основывается на том факте, что даже в худшем случае подграф из $\lceil n^{\frac{\epsilon}{2}}\rceil$ вершин может быть помещен в памяти одной машины со всеми своими ребрами) с большой вероятностью сократиться в $\lceil n^{\frac{\epsilon}{2}}\rceil$ раз, а значит поиск в ширину из всех вершин из $SV(n^{1-\epsilon}\log n)$ нужно будет осуществить на глубину $\lceil \max(n^{1-\frac{3}{2}\epsilon}, 1)\log n\rceil$, что в свою очередь привносит доминирующий вклад в сложностную оценку всего алгоритма.
Обратимся теперь ко второй подзадаче. Чтобы реализовать данный этап,

рассмотрим подграф $G^\epsilon(SV, E^\epsilon)$, где для любых двух вершин $r, t \in SV(n^{1-\epsilon}\log n)$ множество E^ϵ содержит ребро (r, t) тогда и только тогда, когда в ходе предыдущего этапа было установлено, что t достижима из r. На данном подграфе алгоритм может быть запущен рекурсивно:

- Выбрать подмножество $SV^\epsilon(n^{\frac{\epsilon}{2}}\log n)$, и провести параллельный поиск в ширину из его вершин. (В рамках данного поиска, как было отмечено в предыдущем абзаце, путем отождествления вершин подграфа с опорными, внутри подгафа строится гиперграф, позволяющий оптимизировать процедуру поиска)

- Загрузить подграф, образованный из вершин из $SV^\epsilon(n^{\frac{\epsilon}{2}}\log n)$, в память одной машины и провести расчеты локально за один раунд (это возможно, так как даже в худшем случае число ребер в данном графе не превышает $O(n^\epsilon)$)

Замечание: получившийся подграф $G^\epsilon(SV, E^\epsilon)$ может не обладать заданными ограничениями на степень, однако данная проблема может быть легко решена посредством обобщающих подходов, которые будут описаны далее.

3.2. Графы фиксированной степени

Чтобы описать общий подход для решения задачи достижимости, рассмотрим сперва случай, когда каждая вершина графа имеет фиксированные входящую и исходящую степени $\lceil d = n^{\frac{\epsilon}{2}} \rceil$.
В первую очередь мы представим алгоритм для вычислительно наиболее сложной части, а именно - параллельного поиска в ширину из семейства вершин $SV \subset V$. Данный фрагмент может быть изложен обособленно, так как алгоритм не подразумевает какой-либо особой структуры для множества SV, кроме ограничения на количество элементов ($|SV| \in O(n^\epsilon)$).
В качестве параметров алгоритм принимает граф, объем памяти на одной машине, подмножество вершин SV, а так же глубину, на которую необходимо провести поиск. В результате работы алгоритма с большой вероятностью, каждая вершина в графе будет знать все вершины из SV, для которых она находится на расстоянии, не превышающем L.
Замечание: для каждого алгоритма в данной работе мы будем указывать два поля ограничений: ограничения на входные данные и ограничения на параметры AMPC модели.
Покажем теперь корректность **Алгоритма 3.1**, а так же оценим его сложность.

Algorithm 3.1 Поиск в ширину из семейства вершин (G, ϵ, SV, L)

Require: $N = n + m$, $S = n^{\epsilon}$, $\epsilon \in (0, 1)$, $P = \lceil n^{1+\frac{1}{2}\epsilon} \rceil$

Require: $G = G(V, E)$, $\epsilon \in (0, 1)$, $|SV| \in O(n^{\epsilon})$, $\forall v \in V$ $Deg_{out}(v) = Deg_{in}(v) = \lceil n^{\frac{\epsilon}{2}} \rceil$

1: Каждой машине присвоить произвольную вершину, в соответствии с ее номером ▷ **Комментарий:** С каждой вершиной будет ассоциировано $\lceil n^{\frac{\epsilon}{2}} \rceil$ машин

2: Выбрать подмножество $B(n^{\frac{\epsilon}{2}})$

3: Каждую вершину из $V \setminus B(n^{\frac{\epsilon}{2}})$ в графе связать с минимальной по номеру вершиной из $B(n^{\frac{\epsilon}{2}})$ среди ее входящих соседей ▷ Назовем такую вершину $B(v)$

4: В память каждой машины, ассоциированной с конкретной вершиной v, загрузить подграф, образованный всеми вершинами u, такими что $B(u) = B(v)$, и ребрами между ними, а так же всех входящих и исходящих соседей v ▷ **Комментарий:** Все вершины u, такие что $B(u) = B(v)$, обозначим $C(v)$

5: Для каждой вершины инициализируем в распределенном хранилище пустой список R, где будут храниться вершины из SV, из которых v достижима и две таблицы TC и TN с размерностями $(\lceil n^{\frac{\epsilon}{2}} \rceil, \lceil n^{\epsilon} \rceil)$, соответствующие $C(v)$ и множеству входящих соседей v ▷ **Комментарий:** Слова "таблица"и "список"в данном контексте применены условно и используются для облегчения восприятия, на самом деле мы находимся в рамках описанной в начале семантики ключ-значение и используем соответствующие двух- и трех- местные ключи

6: Проинициализировать R для всех вершин из SV самими собой

7: **for** $l = 0$, $l < \lceil \max(\frac{L}{n^{\frac{\epsilon}{2}}}, 1) \log n \rceil$ **do**

8: Для каждой вершины v соответствующие ей $\lceil n^{\frac{\epsilon}{2}} \rceil$ машин копируют $R(v)$ в соответствующие строчки таблиц для всех вершин из $C(v)$

9: Для каждой вершины v соответствующие ей $\lceil \frac{n^{\frac{\epsilon}{2}}}{2} \rceil$ машин строят объединение всех строк из $TC(v)$ и дополняют $R(v)$

10: Для каждой вершины v соответствующие ей $\lceil n^{\frac{\epsilon}{2}} \rceil$ машин копируют $R(v)$ в соответствующие строчки таблиц для всех исходящих соседей v

11: Для каждой вершины v соответствующие ей $\lceil \frac{n^{\frac{\epsilon}{2}}}{2} \rceil$ машин строят объединение всех строк из $TN(v)$ и дополняют $R(v)$

12: **end for**

Ensure: $\forall u \in SV$ and $\forall k \in V$: $dist(u, k) \leq L$ верно: k знает, что достижима из u

Лемма 5. *В результате работы* ***Алгоритма 3.1*** $\forall u \in SV$ *and* $\forall k \in V : dist(u,k) \leq L$ *с большой вероятностью верно: k получит отметку, что достижима из u*

Доказательство.

- **Шаг 3:** Так как $|B(n^{\frac{\epsilon}{2}})| \in O(n^{\frac{\epsilon}{2}} \log n)$ и входящая степень каждой вершины в графе равна $\lceil n^{\frac{\epsilon}{2}} \rceil$, то по **Утверждению 2**, с большой вероятностью или сама вершина или ее входящий сосед принадлежат $B(n^{\frac{\epsilon}{2}})$, то есть данный шаг алгоритма корректен и каждая вершина в графе попадет в ту или иную группу

- **Шаг 7:** Рассмотрим следующий граф: все вершины, попавшие в одну группу в ходе шага три, отождествим с одной гипервершиной, и соединим направленными ребрами все такие гипервершины, составляющие вершины которых связаны хотя бы одним направленным ребром в исходном графе.
  Число гипервершин в таком графе будет равно числу элементов в $B(n^{\frac{\epsilon}{2}} \log n)$, то есть в $\frac{n^{\frac{\epsilon}{2}}}{\log n}$ раз меньше, чем в исходном графе. Таким образом поиск достаточно провести на глубину $\lceil \frac{L \log n}{n^{\frac{\epsilon}{2}}} \rceil$, что и происходит внутри цикла, на **Шаге 7** (сперва инициализируется вся гипервершина, а затем происходит взаимодействие по ребрам между гипервершинами)

□

Лемма 6. *В результате работы* ***Алгоритма 3.1*** *каждая машина делает не более чем* $O(n^{\epsilon})$ *запросов за раунд и использует не более чем* $O(n^{\epsilon})$ *памяти.*

Доказательство.

- **Шаг 3:** Так как входящая степень каждой вершины в графе равна $\lceil n^{\frac{\epsilon}{2}} \rceil$, то просмотреть всех входящих соседей возможно за $O(n^{\frac{\epsilon}{2}})$ запросов

- **Шаг 4:** Так как входящая степень каждой вершины в графе равна $\lceil n^{\frac{\epsilon}{2}} \rceil$, то число вершин, которые будут отождествлены с данной вершиной из $B(n^{\frac{\epsilon}{2}} \log n)$ не превосходит $O(n^{\frac{\epsilon}{2}})$, а значит число ребер между ними принадлежит $O(n^{\epsilon})$

- **Шаг 5:**
  - По условию число вершин в SV не превосходит $O(n^{\epsilon})$
  - Каждую строчку в "таблице"будет заполнять одна машина, а длина строки, в свою очередь, не превосходит n^{ϵ}

- **Шаги 7-12:** Как было отмечено в предыдущем пункте каждая машина обрабатывает одну строку, длина которой ограничена размерами множества SV, то есть не превосходит $O(n^\epsilon)$

□

Лемма 7. ***Алгоритм 3.1*** *завершается через* $O(\max(\frac{L}{n^{\frac{\epsilon}{2}}}, 1)\log^2 n)$ *раундов.*

Доказательство.
- **Шаг 2:** По доказанному ранее сконструировать такое подмножество мы можем за $O(1)$ раундов
- **Шаг 3-6:** Требуют суммарно $O(1)$ раундов
- **Шаг 7:**
  - Шаги 8 и 10 требуют $O(1)$ раундов, так как объем информации для записи (размер множества SV) умещается в один запрос
  - Шаги 9 и 11 требуют $O(\log n)$ раундов каждый, так как число сравниваемых строк на каждом шаге уменьшается в два раза (на первом шаге каждая машина сравнивает две строки и записывает их объединение, таким образом, на втором шаге останется обработать половину от исходного количества строк и т.д)

Наконец, общая сложность цикла равна $O(\max(\frac{L}{n^{\frac{\epsilon}{2}}}, 1)\log^2 n)$

□

Объединим теперь все вышеизложенное в одном утверждении:

Теорема 2. *Пусть дан направленный граф* $G = (V, E)$, $|V| = n$, $|E| = m$. *Пусть так же даны* $\epsilon \in (0, 1)$, $L < n$ *и подмножество* $SV \subset V$, *такое что* $|SV| < n^\epsilon$, *а так же выполняется ограничение на степени:* $\forall v \in V\ Deg_{out}(v) = Deg_{in}(v) = \lceil n^{\frac{\epsilon}{2}} \rceil$. *В таком случае существует вероятностный AMPC алгоритм, который решает задачу поиска в ширину из множества вершин* SV *на глубину* L *за* $O(\max(1, \frac{L}{n^{\frac{\epsilon}{2}}})\log^2 n)$ *раундов, используя объем памяти* n^ϵ *и* $\lceil n^{1+\frac{1}{2}\epsilon} \rceil$ *машин.*

Далее мы приведем сам алгоритм поиска всех вершин, достижимых из конкретной вершины $v \in V$.

Algorithm 3.2 Достижимость из одной вершины (G, v, ϵ)

Require: $N = n + m,\ S = n^{\epsilon},\ \epsilon \in (0, 1),\ P = \lceil n^{1+\frac{1}{2}\epsilon} \rceil$
Require: $G = G(V, E),\ \epsilon \in (0, 1),\ \forall v \in V\ Deg_{out}(v) = Deg_{in}(v) = \lceil n^{\frac{\epsilon}{2}} \rceil$
1: Каждой машине присвоить произвольную вершину, в соответствии с ее номером
2: Выбрать подмножество $SV(n^{1-\epsilon} \log n)$ и включить туда вершину v
3: Запустить **Алгоритм 3.1** $(G, \epsilon, SV(n^{1-\epsilon} \log n), 2\lceil n^{1-\epsilon} \log n \rceil)$
4: Создать пустой граф $G' = G'(SV, NULL)$
5: **for** $v \in SV(n^{1-\epsilon} \log n)$ **do**
6: $\quad \forall u : dist(u, v) \leq \lceil n^{\frac{\epsilon}{2}} \rceil$ записать входящее ребро (u, v) to G'
7: **end for**
8: В графе G' выбрать подмножество $SV'(n^{\frac{\epsilon}{2}})$ и включить туда вершину v
9: Запустить **Алгоритм 3.1** $(G', \epsilon, SV'(n^{\frac{\epsilon}{2}}), 2\lceil n^{\frac{\epsilon}{2}} \rceil)$
10: Локально установить достижимость из v для вершин из $SV'(n^{\frac{\epsilon}{2}})$
Ensure: $\forall u \in V : dist(v, u) \leq \infty$ верно: u знает, что достижима из v

Лемма 8. *В результате работы* ***Алгоритма 3.2*** *с большой вероятностью* $\forall u \in V : dist_(v, u) \leq \infty$ *верно: и знает, что достижима из* v

Доказательство.
- Для вершин из $SV'(n^{\frac{\epsilon}{2}})$ достижимость устанавливается в ходе локальной обработки графа G'
- Для вершин из $SV(n^{1-\epsilon} \log n) \setminus SV'(n^{\frac{\epsilon}{2}})$: По **Утверждению 4** для любой вершины $u \in V$ верно, что если существует путь $P(v, u) : |P| \geq n^{\frac{\epsilon}{2}}$, то с большой вероятностью существует вершина $k \in SV'(n^{\frac{\epsilon}{2}}) : k \in P$ и $dist_P(k, u) \leq 2\lceil n^{\frac{\epsilon}{2}} \rceil$. Таким образом для любой вершины заключение о ее достижимости из вершины v можно сделать, проверив на достижимость из v все такие вершины $k \in SV'(n^{\frac{\epsilon}{2}}) : k \in P$ и $dist_P(k, u) \leq 2\lceil n^{\frac{\epsilon}{2}} \lceil$
- Для вершин из $V \setminus SV(n^{1-\epsilon} \log n)$: Аналогично предыдущему пункту, но поиск нужно провести на глубину $\lceil n^{1-\epsilon} \log n \rceil$

□

Лемма 9. *В результате работы* ***Алгоритма 3.1*** *каждая машина делает не более чем* $O(n^{\epsilon})$ *запросов за раунд и использует не более чем* $O(n^{\epsilon})$ *памяти.*

Доказательство.
- **Шаги 3 и 9:** По **Лемме 6** ограничения на память и число запросов выполняются

- **Шаг 5:** Каждая машина будет добавлять ребра, соответствующие одной вершине, и таких ребер будет не больше, чем элементов в $SV(n^{1-\epsilon}\log n)$, то есть $O(n^\epsilon)$
- **Шаг 10:** Число ребер в графе с $\lceil n^{\frac{\epsilon}{2}} \rceil$ вершинами не превышает $O(n^\epsilon)$

□

Лемма 10. ***Алгоритм 3.2*** *завершается через* $O(n^{1-\frac{3}{2}\epsilon}\log^2 n)$ *раундов.*

Доказательство.
- **Шаг 2:** По доказанному ранее сконструировать такое подмножество мы можем за $O(1)$ раундов путем моделирования случайной величины
- **Шаг 3:** По **Утверждению 7** поиск требует $O(n^{1-\epsilon}\log n\frac{\log n}{n^{\frac{\epsilon}{2}}}) = O(n^{1-\frac{3}{2}\epsilon}\log^2 n)$ раундов
- **Шаг 5:** Может быть завершен за один раунд, так как каждая машина добавляет ребра, соответствующие одной вершине, и их объем не превышает размера одного запроса
- **Шаг 8:** По доказанному ранее сконструировать такое подмножество мы можем за $O(1)$ раундов путем моделирования случайной величины
- **Шаг 9:** По **Утверждению 7** поиск требует $O(\log n)$ раундов
- **Шаг 10:** Может быть завершен за один раунд

□

Теорема 3. *Пусть дан направленный граф* $G = (V, E)$, $|V| = n$, $|E| = m$. *Пусть так же даны* $\epsilon \in (0, 1)$ *и вершина* $v \in V$, *а так же выполняется ограничение на степени:* $\forall v \in V\ Deg_{out}(v)(v) = Deg_{in}(v)(v) = \lceil n^{\frac{\epsilon}{2}} \rceil$. *В таком случае существует вероятностный AMPC алгоритм, который решает задачу поиска всех вершин, достижимых из вершины* v, *за* $O(\max(n^{1-\frac{3}{2}\epsilon}, 1)\log^2 n)$ *раундов, используя объем памяти* n^ϵ *и* $\lceil n^{1+\frac{1}{2}\epsilon} \rceil$ *машин.*

3.3. Обобщение для графов степени меньшей фиксированной

В данной секции мы опишем модернизированную версию алгоритма, позволяющую обрабатывать графы, степень которых меньше $\lceil n^{\frac{\epsilon}{2}} \rceil$.
Основная идея подхода заключается в следующем: в худшем случае (если множества соседей сильно пересекаются) используя $O(n^\epsilon)$ запросов

можно обнаружить как минимум $O(n^{\frac{\epsilon}{2}})$ новых вершин, начав поиск из фиксированной вершины. Таким образом, за $O(1)$ раундов можно увеличить степень каждой вершины до $\lceil n^{\frac{\epsilon}{2}} \rceil$ (аналогично для входящих и исходящих степеней).
Если для какой-то вершины не получилось найти $\lceil n^{\frac{\epsilon}{2}} \rceil$ уникальных вершин, то данная вершина и весь найденный подграф представляют собой набор компонент сильной связности, которые не содержат более никаких вершин извне и могут быть обработаны за один раунд локально. Данный факт следует из наблюдения, что компонента сильной связности представляет собой транзитивное замыкание множеств достижимостей вершин. Однако, полностью подобные вершины из графа выбросить мы не можем, так как это повлечет за собой уменьшение степеней всех остальных вершин. Чтобы устранить данное противоречие мы введем виртуальный конгломерат из $\lceil n^{\frac{\epsilon}{2}} \rceil$ вершин, и связанными с ними ребрами дополним недостающие степени.
Замечание для входящих и исходящих степеней процедура происходит аналогично, за исключением направления ребер.
Дополнительно отметим, что в ходе данной процедуры количество вершин вырастет на $\lceil n^{\frac{\epsilon}{2}} \rceil$, что в свою очередь не меняет доминирующего слагаемого в асимптотике в терминах О-большого. Количество ребер в свою очередь вырастает более значительно, однако в представленных ранее рассуждениях данный параметр фигурировал опосредованно. В данной работе мы не делаем каких либо различий для реберной классификации графов(на плотные или разреженные).

Теорема 4. *Пусть дан направленный граф $G = (V, E)$, $|V| = n$, $|E| = m$. Пусть так же даны $\epsilon \in (0, 1)$ и вершина $v \in V$, а так же выполняется ограничение на степени: $\forall v \in V\ Deg_{out}(v) \leq \lceil n^{\frac{\epsilon}{2}} \rceil$, $Deg_{in}(v) \leq \lceil n^{\frac{\epsilon}{2}} \rceil$. В таком случае существует вероятностный AMPC алгоритм, который решает задачу поиска всех вершин, достижимых из вершины v, за $O(\max(n^{1-\frac{3}{2}\epsilon}, 1) \log^2 n)$ раундов, используя объем памяти n^ϵ и $\lceil n^{1+\frac{1}{2}\epsilon} + n^\epsilon \rceil$ машин.*

Algorithm 3.3 Увеличение степени (G, d)

Require: $G = G(V, E) : \forall v \in V\ Deg_{out}(v) \geq 1,\ Deg_{in}(v) \geq 1$
Ensure: $G' = G'(V', E') : \forall v \in V'\ Deg_{out}(v) \geq \lceil n^{\frac{\epsilon}{2}} \rceil, Deg_{in}(v) \geq \lceil n^{\frac{\epsilon}{2}} \rceil$

1: Каждой машине присвоим произвольную вершину, в соответствии с ее номером
2: **for** $v \in V$ **do**
3: $cIn = Deg_{in}(v)$
4: Начать поиск в ширину из v, увеличивая счетчик cIn при обнаружении новой вершины, прекратить поиск если число запросов превысило $O(n^{\epsilon})$
5: $cOut = Deg_{out}(v)$
6: Начать поиск в ширину из v, увеличивая счетчик $cOut$ при обнаружении новой вершины, прекратить поиск если число запросов превысило $O(n^{\epsilon})$
7: **if** $cOut \leq d$ **then**
8: Добавить недостающее число виртуальных ребер
9: **end if**
10: **if** $cIn \leq d$ **then**
11: Добавить недостающее число виртуальных ребер
12: **end if**
13: **end for**

3.4. Обобщение для произвольного графа

Опишем теперь подход, который позволит перейти к рассмотрению графов со степенями, превышающими $\lceil n^{\frac{\epsilon}{2}} \rceil$.

Каждую вершину, степень которой превышает заданный параметр, мы можем превратить в сбалансированное дерево, степень каждой вершины в котором равна нужной нам величине. Данный метод сделает количество вершин в графе асимптотически близким к числу ребер, однако замечательным фактом является то, что даже при замене всех вершин внутри некоторого пути на подобные деревья, его длина вырастет лишь в константу раз. То есть, таким образом, общая сложность алгоритма поиска не вырастет в терминах О-большого. Тем не менее, важно отметить, что количество машин, которые будут использоваться в данном случае, станет пропорционально числу ребер в рассматриваемом графе. Аналогично сложность такой перестройки не будет превышать $O(1)$ раундов (по количеству уровней в дереве).

Замечание для того, чтобы сохранить условие на минимальные степени вершин в получившемся дереве, всех прямых потомков одного родителя нужно соединить промежуточными ребрами - это увеличит максимальную степень до $O(n^{\frac{\epsilon}{2}})$, что все еще соответствует требованиям, описан-

ным в предыдущих алгоритмах.

Algorithm 3.4 Уменьшение степени (G, d)

Require: $G = G(V, E) : \forall v \in V \; Deg_{out}(v) = Deg_{in}(v) \geq \lceil n^{\frac{\epsilon}{2}} \rceil$
Ensure: $G' = G'(V', E') : \forall v \in V' Deg_{out}(v) = Deg_{in}(v) = \lceil n^{\frac{\epsilon}{2}} \rceil$ или $2\lceil n^{\frac{\epsilon}{2}} \rceil$
  Каждой машине присвоим произвольную вершину, в соответствии с ее номером ▷ С каждой вершиной будет ассоциировано $\lceil n^{1-\epsilon} \rceil$ машин
  for $v \in V$ **do**
    if $Deg_{out}(v) \geq \lceil n^{\frac{\epsilon}{2}} \rceil$ **then**
      $\lceil n^{1-e} \rceil$ машин строят дерево по уровням
    end if
    if $Deg_{in}(v) \leq \lceil n^{\frac{\epsilon}{2}} \rceil$ **then**
      $\lceil n^{1-e} \rceil$ машин строят дерево по уровням
    end if
  end for

Теорема 5. *Пусть дан направленный граф $G = (V, E)$, $|V| = n$, $|E| = m$. Пусть так же даны $\epsilon \in (0, 1)$, $L < n$ и вершина $v \in V$. В таком случае, существует AMPC алгоритм, который решает задачу поиска всех вершин, достижимых из вершины v, за $O(\max(n^{1-\frac{3}{2}\epsilon}, 1) \log^2 n)$ раундов, используя объем памяти n^ϵ и $\lceil m^{1+\frac{1}{2}\epsilon} \rceil$ машин.*

4. Компоненты сильной связности

В заключение, приведем алгоритм поиска компонент сильной связности, который будет в значительной степени опираться на описанную ранее процедуру поиска достижимых вершин. Данный алгоритм был описан в [8] и позволяет вычислить компоненты сильной связности за $O(\log^2 n)$ запросов о достижимости из отдельной вершины.
Ключевая идея алгоритма состоит в разбиении графа на подграфы, которые точно не пересекаются ни по одной из сильносвязных компонент, в свою очередь обозначенные подграфы обрабатываются рекурсивно. Такой подход позволяет более эффективно реализовать параллельное решение задачи. Разбиение на подграфы осуществляется за счет следующего наблюдения:

Теорема 6. *[8] Пусть V - множество всех вершин в графе и $N \subset V$ - некоторое его подмножество, пусть так же вершина v не принадлежит N. Обозначим:*

- *A = вершины, достигнутые из набора вершин N*

- B = *вершины, достигнутые из вершины* v
- C = *вершины, которые достигают* v *и достигаются из* v.

Тогда для следующих подмножеств вершин: $V \setminus (A \cup B), A \setminus B, B \setminus (A \cup C)$ *и* $(A \cap B) \setminus C$ *верно, что никакие вершины из разных подмножеств не лежат в одной компоненте сильной связности.*

Далее в своей работе авторы приводят утверждение, что при определенном выборе вершины v и множества C на каждом рекурсивном шаге, суммарное количество запросов о достижимости не превысит $O(\log^2 n)$. Обобщим теперь все вышесказанное, сформулировав финальное утверждение о сложности поиска компонент сильной связности в AMPC модели. В предыдущих разделах, учитывая рассуждения об общении на графы произвольной степени, мы показали, что существует AMPC алгоритм, который с большой вероятностью решает задачу достижимости из одной вершины за $O(\max(n^{1-\frac{3}{2}\epsilon}, 1)\log^2 n)$ раундов, используя объем памяти n^ϵ и $\lceil m^{1+\frac{1}{2}\epsilon} \rceil$ машин.
Таким образом, подставив данную оценку, в приведенный в [8] алгоритм, получим следующее утверждение:

Теорема 7. *Пусть дан направленный граф* $G = (V, E)$, $|V| = n$, $|E| = m$. *Пусть так же дано* $\epsilon \in (0, 1)$. *В таком случае существует вероятностный AMPC алгоритм, который решает задачу поиска компонент сильной связности за* $O(\max(n^{1-\frac{3}{2}\epsilon}, 1)\log^4 n)$ *раундов, используя* $\lceil m^{1+\frac{1}{2}\epsilon} \rceil$ *машин и объем памяти* n^ϵ.

5. Заключение

В рамках данной работы был получен вероятностный алгоритм поиска компонент сильной связности в графе в AMPC модели. Для описанного подхода были приведены доказательства корректности и сложности, а так же представлены оценки на минимально необходимый объем вычислительных ресурсов(памяти и числа машин). В качестве ключевых особенностей данной реализации можно выделить следующие аспекты: полилогарифмическая или сублинейная сложность при относительно больших объемах локальной памяти, а так же сублинейные по числу вершин(а не ребер) в графе требования к памяти и количеству запросов(что зачастую является наиболее сложным и интересным сценарием).

Distributed strongly connected components search in an adaptive model
Moldovanov I.V.

To study the efficiency of distributed algorithms, a number of mathematical formalizations with new concepts of algorithm complexity are introduced. In this paper complexity of finding strongly connected components in the AMPC model is investigated. AMPC model, unlike other more limited distributed formalizations, allows to build a query tree within one step of the algorithm. A probabilistic algorithm is obtained that implements strongly connected components search in polylogarithmic or sublinear time (depending on the amount of available local memory). The amount of required local memory is sublinear with respect to the number of vertices in the graph.

Keywords: distributed algorithms, probabilistic algorithms, strongly connected components.

References

[1] Grzegorz M.M., Austern Aart J.C Bik James C. Dehnert Ilan Horn Naty Leiser Grzegorz Czajkowski, "Pregel: a system for large-scale graph processing", *Proceedings of the 2010 international conference on Management of data, ACM*, 2010, 135–146.

[2] Howard Karloff, Siddharth Suri, and Sergei Vassilvitskii, "A model of computation for mapreduce.", *In Proceedings of the twenty-first annual ACM-SIAM symposium on Discrete Algorithms, SIAM*, 2010, 938—948.

[3] Soheil Behnezhad, Laxman Dhulipala, Hossein Esfandiari, "Massively Parallel Computation via Remote Memory Access.", *The 31st ACM Symposium on Parallelism in Algorithms and Architectures*, 2019.

[4] Soheil Behnezhad, Laxman Dhulipala, Hossein Esfandiari, "Parallel graph algorithms in constant adaptive rounds: theory meets practice.", *Proceedings of the VLDB Endowment*, **13** (2020).

[5] Tarjan R. E., "Depth-first search and linear graph algorithms.", *SIAM Journal on Computing.*, **1** (1972), 146—160.

[6] Micha Sharir., "A strong-connectivity algorithm and its applications to data flow analysis.", *Computers and Mathematics with Applications.*, **7** (1981), 67—72.

[7] Fleischer, Lisa K.; Hendrickson, Bruce; Pınar, Ali, "On Identifying Strongly Connected Components in Parallel.", *Parallel and Distributed Processing, Lecture Notes in Computer Science.*, **1800** (2000), 505—511.

[8] Warren Schudy, "Finding strongly connected components in parallel using $O(\log^2 n)$ reachability queries.", *SPAA '08: Proceedings of*

the twentieth annual symposium on Parallelism in algorithms and architectures., 2008, 146-–151.

[9] J. D. Ullman and M. Yannakakis., "High probability parallel transitive-closure algorithms.", *SIAM J.Comput.*, **20(1)** (1991), 100—125.

[10] Alexandr Andoni, Zhao Song, Clifford Stein, "Parallel graph connectivity in log diameter rounds.", *59th IEEE Annual Symposiumon Foundations of Computer Science, FOCS.*, 2018, 674—685.

[11] Chernoff, Herman, "A Measure of Asymptotic Efficiency for Tests of a Hypothesis Based on the sum of Observations.", *The Annals of Mathematical Statistics.*, **23(4)** (1952), 493—507.

Угол между плоскостями линейных тестовых алгоритмов

М. В. Носов[1]

В работе исследуется вопрос об угле между нормальными векторами разделяющих гиперплоскостей, задаваемых линейными тестовыми алгоритмами. Вопрос рассматривается для различных опорных множеств тестов.

Ключевые слова: линейные тестовые алгоритмы, разделяющие гиперплоскости, тупиковые тесты.

В теории тестовых алгоритмов рапознавания выделяются три линейных алгоритма, исследуем вопрос о величине угла между соответствующими гиперплоскостями. Пусть E^n - n-мерный единичный куб, (T_0, T_1) - материал обучения, $T_0, T_1 \subset E^n, T_0 \bigcap T_1 = \emptyset, T_0 \neq \emptyset, T_1 \neq \emptyset$, пусть T - опорное множество тестов.

Первый алгоритм - линейный тестовый алгоритм[1][3], задаваемый решающим правилом $R_1(t), t = (t_1, \ldots, t_n)$

$$R_1(t) = \sum_{i=1}^{n} p_i t_i + p_0,$$

$$p_i = \frac{1}{|T|} \sum_{\tau \in T} \tau_i, i = 1, \ldots, n,$$

$p = (p_1, \ldots, p_n)$ - вектор информационных весов,

p_0 - порог - действительное число.

Второй алгоритм - линейный тестовый алгоритм Журавлёва Ю.И.[2], решающее правило имеет вид

$$R_2(t) = 2 \sum_{i=1}^{n} p_i \Delta s_i t_i - \sum_{i=1}^{n} p_i \Delta s_i,$$

$$\Delta s_i = s_{1i} - s_{0i},$$

$s_{1i} = \frac{1}{|T_1|} \sum_{\theta \in T_1} \theta_i$ - центр тяжести T_1,

$s_{0i} = \frac{1}{|T_0|} \sum_{\theta \in T_0} \theta_i$ - центр тяжести T_0,

[1] *Носов Михаил Васильевич* — с.н.с. каф. математической теории интеллектуальных систем мех.-мат. ф-та МГУ, e-mail: mvnosov@rambler.ru.

Nosov Michail Vasilevich-senior researcher, Lomonosov Moscow State University, Faculty of Mechanics and Mathematics, Chair of Mathematical Theory of Intellectual Systems.

$$\theta = (\theta_1, \ldots, \theta_n),$$
$$\Delta s_i = s_{1i} - s_{0i},$$
$$i = 1, \ldots, n,$$
$$\Delta s = (\Delta s_1, \ldots, \Delta s_n).$$

Третий тестовый алгоритм - линейный тестовый алгоритм - наилучшее линейное приближение алгоритма голосования Кудрявцева В.Б.[1][3], его решающее правило имеет вид

$$R_3(t) = 4\sum_{i=1}^{n} q_i \Delta s_i t_i - 2\sum_{i=1}^{n} q_i \Delta s_i,$$

$$q_i = \sum_{\tau \in T} \frac{\tau_i}{2^{|\tau|}}, i = 1, \ldots, n.$$

Таким образом, имеется три нормальных вектора

$$\omega_1 = (p_1, \ldots, p_n),$$
$$\omega_2 = (p_1 \Delta s_1, \ldots, p_n \Delta s_n),$$
$$\omega_3 = (q_1 \Delta s_1, \ldots, q_n \Delta s_n).$$

Оценим угол между этими векторами для некоторых случаев материала обучения (T_0, T_1) и опорного множества тестов T.

1.Пусть $T_0 = \{(0, \ldots, 0)\}, T_1 = E^{n,k}$ - k-ый слой куба, $1 \leq k \leq n$, в качестве опорного множества возьмём все тупиковые тесты, очевидно, $T = E^{n,n-k+1}$. Ввиду "равноправия"всех координат все три вектора сонаправлены.

2.Пусть n - чётное, $n = 2l$. Пусть T_0 состоит из одного вектора, у которого первые l координат 1, остальные 0:

$$T_0 = \{(1, \ldots, 1, 0, \ldots, 0)\},$$

условно можно написать

$$T_0 = \{(0, \ldots, 0)\} \oplus (1, \ldots, 1, 0, \ldots, 0).$$

T_1 представить в виде

$$T_1 = E^{n,k} \oplus (1, \ldots, 1, 0, \ldots, 0).$$

Очевидно, что $T = E^{n,n-k+1}$ и вектор p сонаправлен единичному. При $1 \leq i \leq l, s_{0i} = 1$,

$$s_{1i} = \frac{1}{\binom{n}{k}} \sum_{\theta \in T_1} \theta_i = \frac{1}{\binom{n}{k}} \sum_{\theta \in E^{n,k}} (\theta_i \oplus 1) = \frac{1}{\binom{n}{k}} \sum_{\theta \in E^{n,k}} (1 - \theta_i) =$$

$$= 1 - \frac{1}{\binom{n}{k}}\binom{n-1}{k-1} = 1 - \frac{k}{n(n-k+1)},$$

$$\Delta s_i = -\frac{k}{n(n-k+1)}.$$

При $l \le i \le 2l, s_{0i} = 0$,

$$s_{1i} = \frac{k}{n(n-k+1)},$$

$$\Delta s_i = \frac{k}{n(n-k+1)}.$$

Следовательно, у вектора Δs все координаты одинаковы по абсолютной величине, но в первой половине отрицательны, а во второй положительны. Значит вектор w_1 перпендикулярен вектору w_2 и вектору w_3.

3.Перпендикулярность векторов в предыдущем случае основывалась на положительности половины координат и отрицательности другой половины для вектора, соединяющего центры тяжестей T_0 и T_1. Всегда можно поменять в признаке 0 и 1, это не приведёт к изменению опорного множества, но все координаты вектора Δs будут одного знака. Итак, $\Delta s \ge 0$; ясно, что $1/|T| \le p_i \le 1$ для ненулевых p_i и пусть хотя бы в одной координате вектора p и Δs положительны.Тогда имеем

$$\cos(\widehat{w_1, w_2}) = \frac{\sum\limits_{i=1}^{n} p_i^2 \Delta s_i}{\sqrt{\sum\limits_{i=1}^{n} p_i^2 \Delta s_i^2}\sqrt{\sum\limits_{i=1}^{n} p_i^2}} \ge \frac{\sum\limits_{i=1}^{n} p_i^2 \Delta s_i}{\sqrt{\sum\limits_{i=1, p_i \ne 0} \Delta s_i^2}\sqrt{n}} \ge$$

$$\ge \frac{\frac{1}{|T|^2}\sum\limits_{i=1, p_i \ne 0} \Delta s_i}{\sqrt{\sum\limits_{i=1, p_i \ne 0} \Delta s_i^2}\sqrt{n}} \ge \frac{1}{|T|^2\sqrt{n}},$$

$$(\widehat{w_1, w_2}) \le \arccos \frac{1}{|T|^2\sqrt{n}}. \qquad (1)$$

Правую часть неравенства можно увеличить, доведя до соотношения от n

$$(\widehat{w_1, w_2}) \le \arccos \frac{1}{(\binom{n}{[n/2]})^2\sqrt{n}}.$$

Далее, при $p_i > 0$

$$\frac{|T|}{2^{\max|\tau|}} p_i \le \sum_{\tau \in T} \frac{\tau_i}{2^{|\tau|}} \le \frac{|T|}{2^{\min|\tau|}} p_i,$$

$$\cos(\widehat{w_1, w_3}) = \frac{\sum\limits_{i=1}^{n} p_i q_i \Delta s_i}{\sqrt{\sum\limits_{i=1}^{n} q_i^2 \Delta s_i^2}\sqrt{\sum\limits_{i=1}^{n} p_i^2}} \geq \frac{\frac{|T|}{2^{\max|\tau|}} \sum\limits_{i=1}^{n} p_i^2 \Delta s_i}{\frac{|T|}{2^{\min|\tau|}}\sqrt{\sum\limits_{i=1}^{n} p_i^2 \Delta s_i^2}\sqrt{\sum\limits_{i=1}^{n} p_i^2}} \geq$$

$$\geq \frac{1}{2^{\max|\tau| - \min|\tau|}|T|^2\sqrt{n}},$$

$$(\widehat{w_1, w_3}) \leq \arccos \frac{1}{2^{\max|\tau| - \min|\tau|}|T|^2\sqrt{n}}. \qquad (2)$$

Доводим соотношение до n

$$(\widehat{w_1, w_3}) \leq \arccos \frac{1}{2^{n-1}(\binom{n}{[n/2]})^2\sqrt{n}}.$$

Пример. Пусть $T_0 = \{(0, \ldots, 0)\}, T_1 = \{(1, 1, 0, 0, \ldots, 0),$ $(1, 0, 1, 0, \ldots, 0), \ldots, (1, 0, 0, 0, \ldots, 1)\}$. Очевидно, множество тупиковых тестов $T = \{(1, 0, \ldots, 0), (0, 1, \ldots, 1)\}$. Тогда

$$w_1 = p = (\frac{1}{2}, \ldots, \frac{1}{2}), q = (\frac{1}{2}, \frac{1}{2^{n-1}}, \ldots, \frac{1}{2^{n-1}}),$$

$$s_1 = (1, \frac{1}{n-1}, \frac{1}{n-1}, \ldots, \frac{1}{n-1}), s_0 = (0, \ldots, 0),$$

$$\Delta s = (1, \frac{1}{n-1}, \frac{1}{n-1}, \ldots, \frac{1}{n-1}),$$

$$w_2 = p\Delta s = (\frac{1}{2}, \frac{1}{2(n-1)}, \frac{1}{2(n-1)}, \ldots, \frac{1}{2(n-1)}),$$

$$w_3 = q\Delta s = (\frac{1}{2}, \frac{1}{2^{n-1}(n-1)}, \frac{1}{2^{n-1}(n-1)}, \ldots, \frac{1}{2^{n-1}(n-1)}).$$

Из неравенства (1) следует

$$(\widehat{w_1, w_2}) \leq \arccos \frac{1}{4\sqrt{n}},$$

Точное значение

$$(\widehat{w_1, w_2}) = \arccos \frac{2}{\sqrt{n}\sqrt{1 + \frac{1}{n-1}}}.$$

Из неравенства (2) следует

$$(\widehat{w_1, w_3}) \leq \arccos \frac{1}{2^n\sqrt{n}},$$

Точное значение

$$(\widehat{w_1, w_3}) = \arccos \frac{1 + \frac{1}{2^{n-1}}}{\sqrt{n}\sqrt{1 + \frac{1}{2^{2n-4}(n-1)}}}.$$

4.Исследуем угол между w_2 и w_3 в общем случае

$$\cos(\widehat{w_2, w_3}) = \frac{\sum\limits_{i=1}^{n} p_i q_i \Delta s_i^2}{\sqrt{\sum\limits_{i=1}^{n} q_i^2 \Delta s_i^2}\sqrt{\sum\limits_{i=1}^{n} p_i^2 \Delta s_i^2}} \geq \frac{\frac{|T|}{2^{\max|\tau|}} \sum\limits_{i=1}^{n} p_i^2 \Delta s_i^2}{\frac{|T|}{2^{\min|\tau|}}\sqrt{\sum\limits_{i=1}^{n} p_i^2 \Delta s_i^2}\sqrt{\sum\limits_{i=1}^{n} p_i^2 \Delta s_i^2}} =$$

$$= \frac{1}{2^{\max|\tau| - \min|\tau|}},$$

$$(\widehat{w_2, w_3}) \leq \arccos \frac{1}{2^{\max|\tau| - \min|\tau|}}.$$

Доводим соотношение до n

$$(\widehat{w_2, w_3}) \leq \arccos \frac{1}{2^{n-1}}.$$

5. Используем результат Коршунова А.Д. для класса монотонных функций $M(n)$.

Лемма [4]. У почти всех функций из $M(n)$ нижние единицы расположены в $E^{n,n/2-1}, E^{n,n/2}, E^{n,n/2+1}$ при чётном n и
в $E^{n,(n-3)/2}, E^{n,(n-1)/2}, E^{n,(n+1)/2}, E^{n,(n+3)/2}$ при нечётном n.

Из выражения $\cos(\widehat{w_2, w_3})$ видно, что можно считать все Δs_i неотрицательными.

Рассмотрим чётный случай, $n = 2l$. В качестве T берём тупиковые тесты, лежащие в $E^{n,n/2-1}, E^{n,n/2}, E^{n,n/2+1}$. Введём обозначения

$$a_i = |\{\tau \in T | \tau_i = 1, |\tau| = l - 1\}|,$$
$$b_i = |\{\tau \in T | \tau_i = 1, |\tau| = l\}|,$$
$$c_i = |\{\tau \in T | \tau_i = 1, |\tau| = l + 1\}|.$$

Тогда

$$|T|p = (\ldots, a_i + b_i + c_i, \ldots), 2^l q = (\ldots, 2a_i + b_i + \frac{c_i}{2}, \ldots),$$

$$2^l q = |T|p + z, z = (\ldots, a_i - \frac{c_i}{2}, \ldots).$$

Имеет место следующее неравенство

$$-\frac{1}{2}|T|p_i \le z_i \le |T|p_i, i = 1, \dots, n.$$

Угол между векторами w_2 и w_3 равен углу между векторами u_2 и u_3

$$u_2 = (p_1|T|\Delta s_1, \dots, p_n|T|\Delta s_n),$$

$$u_3 = (2^l q_1 \Delta s_1, \dots, 2^l q_n \Delta s_n).$$

Далее

$$u_3 = u_2 + (z_1|\Delta s_1|, \dots, z_n|\Delta s_n|),$$

как было показано

$$-\frac{1}{2}|T|p_i|\Delta s_i| \le z_i|\Delta s_i| \le |T|p_i|\Delta s_i|,$$

таким образом, угол между векторами u_2 u_3 не превышает угла между вектором u_2 и вектором с концом в вершине параллелепипеда

$$\prod_{i=1}^{n}[-\frac{1}{2}|T|p_i|\Delta s_i|, |T|p_i|\Delta s_i|] + u_2 =$$

$$= \prod_{i=1}^{n}[\frac{1}{2}|T|p_i|\Delta s_i|, 2|T|p_i|\Delta s_i|].$$

Достаточно очевидно, что максимум величины угла достигается на одной из вершин $(2^{g_1}|T|_1\Delta s_1, \dots, 2^{g_n}|T|_1\Delta s_n)$, пусть $g_i = -1, i \in J_1, g_i = 1, i \in J_2, J_1, J_2 \subset \{1, \dots, n\}$. Проведём через эту вершину гиперплоскость с направляющим вектором u_2, её уравнение имеет вид

$$\sum_{i=1}^{n} p_i|T||\Delta s_i|t_i - \frac{1}{2}\sum_{i\in J_1} p_i^2|T|^2|\Delta s_i|^2 - 2\sum_{i\in J_2} p_i^2|T|^2|\Delta s_i|^2 = 0,$$

эта гиперплоскость пересекает прямую, проходящую через начало координат, с направляющим вектором u_2 в точке λu_2, где λ удовлетворяет уравнению

$$\sum_{i=1}^{n} p_i^2|T|^2|\Delta s_i|^2\lambda - \frac{1}{2}\sum_{i\in J_1} p_i^2|T|^2|\Delta s_i|^2 - 2\sum_{i\in J_2} p_i^2|T|^2|\Delta s_i|^2 = 0,$$

Пусть α - угол между вектором u_2 и вектором с началом в начале координат и концом в точке $(2^{g_1}|T|_1\Delta s_1, \dots, 2^{g_n}|T|_1\Delta s_n)$, тогда

$$\tan^2\alpha = \frac{\sum\limits_{i\in J_1}(\frac{1}{2}-\lambda)^2 p_i^2|T|^2|\Delta s_i|^2 + \sum\limits_{i\in J_2}(2-\lambda)^2 p_i^2|T|^2|\Delta s_i|^2}{\lambda^2\sum\limits_{i=1}^{n} p_i^2|T|^2|\Delta s_i|^2}.$$

Пусть

$$P_1 = \sum_{i \in J_1} p_i^2 |T|^2 |\Delta s_i|^2, P_2 = \sum_{i \in J_2} p_i^2 |T|^2 |\Delta s_i|^2,$$

Тогда

$$\operatorname{tan}^2 \alpha = \frac{(\frac{1}{2} - \lambda)^2 P_1 + (2 - \lambda)^2 P_2}{\lambda^2 (P_1 + P_2)},$$

$$\lambda = \frac{\frac{1}{2} P_1 + 2 P_2}{P_1 + P_2}.$$

Несложно найти максимальное значение $\operatorname{tan} \alpha$ равное $\frac{3}{4}$. Таким образом, при чётном n и векторах опорного множества, лежащих во множествах $E^{n,n/2-1}, E^{n,n/2}, E^{n,n/2+1}$, угол между векторами w_2 и w_3 не превышает $\arctan \frac{3}{4}$, $(\arctan \frac{3}{4} = 36,87^\circ)$.

Рассмотрим нечётный случай, $n = 2l+1$ аналогично чётному. В качестве T берём тупиковые тесты, лежащие в $E^{n,(n-3)/2}$, $E^{n,(n-1)/2}$, $E^{n,(n+1)/2}$, $E^{n,(n+3)/2}$. Введём обозначения

$$a_i = |\{\tau \in T | \tau_i = 1, |\tau| = l - 1\}|,$$
$$b_i = |\{\tau \in T | \tau_i = 1, |\tau| = l\}|,$$
$$c_i = |\{\tau \in T | \tau_i = 1, |\tau| = l + 1\}|,$$
$$d_i = |\{\tau \in T | \tau_i = 1, |\tau| = l + 2\}|.$$

Тогда

$$|T|p = (\ldots, a_i + b_i + c_i + d_i, \ldots),$$
$$2^{l+\frac{1}{2}} q = (\ldots, 2^{\frac{3}{2}} a_i + 2^{\frac{1}{2}} b_i + 2^{-\frac{1}{2}} c_i + 2^{-\frac{3}{2}} d_i, \ldots),$$
$$2^{l+\frac{1}{2}} q = |T|p + z,$$
$$z = (\ldots, (2\sqrt{2} - 1) a_i + (\sqrt{2} - 1) b_i - (1 - \frac{1}{\sqrt{2}}) c_i - (1 - \frac{1}{2\sqrt{2}}) d_i, \ldots).$$

Имеет место следующее неравенство

$$-\frac{2\sqrt{2} - 1}{2\sqrt{2}} |T| p_i \leq z_i \leq (2\sqrt{2} - 1) |T| p_i, i = 1, \ldots, n.$$

Угол между векторами w_2 и w_3 равен углу между векторами u_2 и u_3

$$u_2 = (p_1 |T| \Delta s_1, \ldots, p_n |T| \Delta s_n),$$
$$u_3 = (2^{l+\frac{1}{2}} q_1 \Delta s_1, \ldots, 2^{l+\frac{1}{2}} q_n \Delta s_n).$$

Далее производя вычисления, как в чётном случае приходим к результату.При нечётном n и векторах опорного множества, лежащих во множествах $E^{n,(n-3)/2}, E^{n,(n-1)/2}, E^{n,(n+1)/2}, E^{n,(n+3)/2}$, угол между векторами w_2 и w_3 не превышает $\arctan \frac{\sqrt{7\sqrt{2}}}{4}$, $(\arctan \frac{\sqrt{7\sqrt{2}}}{4} = 38,19^\circ)$.

Список литературы

[1] Константинов Р.М., Королёва З.Е., Кудрявцев В.Б. Комбинаторно-логический подход к задачам прогноза рудоносности Проблемы кибернетики 1976 31 5–33

[2] Дмитриев А.Н., Журавлёв Ю.И., Кренделёв Ф.П. О математических принципах классификации предметов и явлений Дискретный анализ 1966 7 3–15

[3] Алешин С.В. Распознавание динамических образов Издательство Московского университета Москва 1996 97

[4] Коршунов А.Д. О числе монотонных булевых функций Проблемы кибернетики 1981 38 5–108

The angle between the planes of linear test algorithms
Nosov M.V.

The paper investigates the question of the angle between the normal vectors of separating hyperplanes defined by linear test algorithms. The question is considered for various reference sets of tests.

Keywords: linear test algorithms separating hyperplanes, dead-end tests.

References

[1] Konstantinov R.M., Koroleva Z.E., Kudryavtsev V.B., "Combinatorial-logical approach to the problems of ore-bearing prediction", *Problems of Cybernetics*, **31** (1976), 5–33 (In Russian).

[2] Dmitriev A.N., Zhuravlev Yu.I., Krendelev F.P., "On mathematical principles of classification of objects and phenomena", *Discrete analysis*, **7** (1966), 3–15 (In Russian).

[3] Aleshin S.V., *Dynamic image recognition*, Moscow University Press, Moscow, 1996 (In Russian), 97 c.

[4] Korshunov A.D., "On the number of monotone Boolean functions", *Problems of Cybernetics*, **38** (1981), 5–108 (In Russian)

Часть 3.
Математические модели

Задача K-конечнопорожденности для предполных классов линейных автоматов, составляющих A-критериальную систему в пространстве линейных автоматов.

В. А. Бирюкова[1]

В данной статье рассматривается проблема K- и A-конечнопорожденности для предполных классов линейных автоматов, функционирующих над полем Галуа, состоящим из двух элементов. Для каждого исследуемого класса был предъявлен конечный базис. Совокупность исследуемых классов составляет A-критериальную систему в классе линейных автоматов.

Ключевые слова: конечный автомат, линейный автомат, операции композиции, обратная связь, полнота, замкнутый класс, предполный класс, K-конечнопорожденный класс, A-конечнопорожденный класс.

1. Введение

С середины прошлого века вплоть до настоящего времени не уменьшается интерес к изучению различных свойств конечных автоматов. Конечный автомат можно представить как устройство, в каждый дискретный момент времени пребывающее в одном из конечного числа состояний и обладающее входом и выходом [1]. Описать действие автомата можно с помощью функций k-значной логики. Труды по данной тематике принадлежат таким видным ученым как С.К. Клини [2], Э.Ф. Муру [3], Дж. фон-Нейману [4] и др. В рамках отечественной школы кибернетики начало положили такие известные ученые, как С.В. Яблонский [5], О.Б. Лупанов [6] и В.Б. Кудрявцев [7],[8].

Интересной задачей является изучение вопросов, рассмотренных как для всего пространства конечных автоматов, так и для его различных подмножеств. В частности, задачи полноты и конечнопорожденности для подклассов конечных автоматов, изучение структуры данных подклассов - например, какие предполные или замкнутые классы есть и т.п.

[1] *Бирюкова Вероника Андреевна* — аспирант кафедры Математической теории интеллектуальных систем механико-математического факультета МГУ, e-mail: biryukovaveronika@mail.ru.

Biryukova Veronika Andreevna — graduate student, Lomonosov Moscow State University, Faculty of Mechanics and Mathematics, Chair of Mathematical Theory of Intellectual Systems.

В работах А.А. Часовских [9]-[12] подробно рассматриваются вопросы полноты и выразимости в классе линейных автоматов над произвольными конечными полями. Проблема линейной реализуемости автоматов — представления линейными функциями автоматов, функционирующих над конечными полями, — излагается, в частности, в книге А. Гилла "Линейные последовательностные машины"[13].

Естественным продолжением работ в этом направлении является изучение вопросов конечнопорожденности относительно разных множеств операций для различных классов линейных автоматов. В представленной работе будут решены задачи K- и A- конечнопорожденности для предполных классов линейных автоматов, совокупность которых представляет собой A-критериальную систему в пространстве линейных автоматов.

2. Основные понятия

С помощью шестерки $V = (A, Q, B, \varphi, \psi, q_0)$ можно задать инициальный абстрактный конечный автомат [1].Обозначим поле Галуа, состоящее из k элементов E_k. Если существуют натуральные числа $n, s \in \mathbb{N}$ такие, что $A = E_k^n,\ Q = E_k^s,\ B = E_k$ и φ, ψ являются линейными операторами k-значной логики, то автомат V является линейным автоматом.

В данной работе рассматриваются линейные автоматы над полем E_2. Мы будем применять к автоматам операции композиции, определенные в соответствии с работой [1]. Равными автоматами мы называем автоматы, задающие на существенных переменных равные ограниченно-детерминированные функции и отличающиеся только множеством (возможно пустым) фиктивных переменных.

Таким образом, мы будем рассматривать операции переименования переменных, отождествления переменных, подстановки одного автомата на вход другого автомата и операции обратной связи [1] как операции композиции.

Через $\mathcal{B}$ обозначим множество автоматов, состоящее из сумматора по модулю два, инвертора и задержки с нулевым начальным состоянием.

Линейные автоматы над E_2 получаем из множества $\mathcal{B}$, используя замыкание по операциям композиции.

Обозначим L_2 множество всех линейных автоматов над полем E_2.

Пусть $M \subseteq L_2$ — некоторое подмножество линейных автоматов. Будем называть K-замыканием $K(M)$ множество всех линейных автоматов, полученных с помощью операций композиции из элементов множества M.

Если $M \subseteq L_2,\ K(M) = M$, то M является K-замкнутым классом.

Если $M \subseteq L_2,\ K(M) = L_2$, то M является K-полным множеством.

Если $M \subset L_2,\ M \neq L_2,\ K(M) = M,\ \forall\ f \in L_2 \backslash M,\ K(\{f\} \cup M) = L_2$, то M есть K-предполный класс.

K-замкнутый класс M называется K-конечнопорожденным, если $\exists\ \{f_1, \ldots, f_m\} \subseteq M :\ M = K(\{f_1, \ldots, f_m\})$.

Также можно рассмотреть в пространстве L_2 оператор A-замыкания — аппроксимационного замыкания. Для этого введем несколько вспомогательных определений.

Пусть $f(x_1, \ldots, x_n) \in L_2,\ M \subseteq L_2,\ \tau \in \mathbb{Z}_+$.

Если существует $g(x_1, \ldots, x_n) \in K(M)$ такой, что для любых входных последовательностей $\alpha_i = a_i(0)a_i(1)\ldots,\quad i = \overline{1,n}$ автоматы f и g будут выдавать последовательности, совпадающие на первых τ элементах (обозначим подобное свойство так: $f(\alpha_1, \ldots, \alpha_n) \stackrel{\tau}{=} g(\alpha_1, \ldots, \alpha_n),$) то говорим, что f — τ-выразима через M.

Если f — τ-выразима через M для любого $\tau \in \mathbb{Z}_+$, то f — выразима через M.

Определим понятие оператора аппроксимационного замыкания множества M. Оператор A-замыкания сопоставляет множеству автоматов M множество всех автоматов, выразимым через M, т.е. $A(M) = \left\{ f \;\middle|\; f \in L_2,\ f \text{ — выразим через } M \right\}$.

Аналогично оператору K-замыкания для аппроксимационного замыкания вводятся понятия замкнутости, полноты, предполноты и конечнопорожденности.

Если $A(M) = M$, то M является A-замкнутым классом.

Если $A(M) = L_2$, то M есть A-полное множество.

Если $M \subset L_2,\ M \neq L_2,\ K(M) = M,\ \forall\ f \in L_2 \backslash M,\ A(\{f\} \cup M) = L_2$, то M — A-предполный класс.

A-замкнутый класс M называется A-конечнопорожденным, если $\exists\ \{f_1, \ldots, f_m\} \subseteq M : M = A(\{f_1, \ldots, f_m\})$.

Введем для формальных степенных рядов над полем E_2 следующее обозначение:

$R_2(\xi) = \left\{ \sum\limits_{t=0}^{\infty} a(t)\xi^t \;\middle|\; a(0), \ldots, a(t), \ldots \in E_2^{\infty} \right\}$ – множество формальных степенных рядов переменной ξ с коэффициентами из поля E_2.

$E_2^{\infty} = \left\{ a(0), \ldots, a(t), \ldots \;\middle|\; t \in \mathbb{N}, \forall\ t\ a(t) \in E_2 \right\}$ — это множество бесконечных последовательностей элементов поля E_2.

$E_2'(\xi) = \left\{ \sum\limits_{t=0}^{\infty} a(t)\xi^t \ \mid a(0), \ldots, a(t), \ldots \in E_2^{\infty}$— периодическая (с предпериодом) последовательность $\right\} \subset R_2(\xi)$.

Также для $E_2'(\xi)$ есть эквивалентное определение [9] через многочлены от переменной ξ:

$$E_2'(\xi) = \left\{ \frac{u}{v} \mid u, v \in E_2[\xi],\ v = 1 + \xi \cdot v', v' \in E_2[\xi] \right\} = \left\{ \frac{u}{v} \mid u, v \in E_2[\xi],\ v(0) = 1 \right\}.$$

Таким образом, $E_2^{'}(\xi)$ является подкольцом поля отношений $E_2(\xi)$ [14].

Автомат f можно рассматривать как преобразователь формальных рядов:

$$f(x_1, \ldots, x_n) : (R(\xi)_2)^n \to R_2(\xi), \tag{3}$$

где x_i принимают значения из $R_2(\xi)\ \ \forall\, i = \overline{1, n}$.

В работе [9] доказано, что любой линейный автомат может быть представлен следующим образом:

$$f(x_1, \ldots, x_n) = \sum_{i=1}^{n} \mu_i x_i + \mu_0, \tag{4}$$

где $n \in \mathbb{N}$, $\mu_j \in E_2^{'}(\xi)\ \forall\, j = \overline{0, n}$, x_i принимают значения из $R_2(\xi)\ \ \forall\, i = \overline{1, n}$.

И наоборот, любой автомат $f(x_1, \ldots, x_n)$, представимый в виде (4), является линейным.

Введем обозначение для множества коэффициентов μ_i при переменных автомата f: $U(f) = \{\mu_1, \ldots, \mu_n\}$.

Не трудно заметить, что переменная x_i является существенной переменной линейного автомата f, если $\mu_i \neq 0$. Переменная x_i называется непосредственной переменной линейного автомата f, если $\mu_i(0) = 1$.

Пусть линейный автомат f задан равенством (4). Рассмотрим, как применение операций композиции выражается в терминах представления (4).

1) Переименование переменных.

$$f(x_1, \ldots, x_{n-1}, x_n) = \sum_{i=1}^{n} \mu_i x_i + \mu_0 = \sum_{i=1}^{n} \mu_i \tilde{x_i} + \mu_0 = f(\tilde{x}_1, \ldots, \tilde{x}_{n-1}, \tilde{x}_n). \tag{5}$$

2) Отождествление переменных. Без ограничения общности пусть у линейного автомата $f(x_1,\ldots,x_n)$ отождествлены переменные x_{n-1} и x_n. Тогда

$$f(x_1,\ldots,x_{n-1},x_{n-1}) = g(x_1,\ldots,x_{n-1}) = \sum_{i=1}^{n-2}\mu_i x_i + \big(\mu_{n-1}+\mu_n\big)x_{n-1} + \mu_0. \tag{6}$$

Таким образом, отождествление переменных x_{n-1} и x_n линейного автомата f приводит к сложению коэффициентов μ_{n-1} и μ_n.

3) Подстановка одного автомата на вход другого автомата. Пусть есть линейные автоматы $f(x_1,\ldots,x_n)$ и $h(x'_1,\ldots,x'_m)$. Без ограничения общности пусть автомат h подставляется на n-ый вход автомата f. Тогда:

$$f(x_1,\ldots,x_n) = \sum_{i=1}^{n}\mu_i x_i + \mu_0, \tag{7}$$

$$h(x'_1,\ldots,x'_m) = \sum_{i=1}^{m}\mu'_i x'_i + \mu'_0, \tag{8}$$

$$g(x_1,\ldots,x_{n-1},x'_1,\ldots,x'_m) = f(x_1,\ldots,x_{n-1},h(x'_1,\ldots,x'_m)), \tag{9}$$

$$g(x_1,\ldots,x_{n-1},x'_1,\ldots,x'_m) = \sum_{i=1}^{n-1}\mu_i x_i + \mu_0 + \mu_n\cdot\left(\sum_{i=1}^{m}\mu'_i x'_i + \mu'_0\right) =$$

$$= \sum_{i=1}^{n-1}\mu_i x_i + \sum_{i=1}^{m}\mu_n\mu'_i x'_i + \big(\mu_n\mu'_0 + \mu_0\big). \tag{10}$$

Как видно из выражения (10), операция подстановки линейного автомата h на n-ый вход автомата f приводит к умножению коэффициентов автомата h на коэффициент μ_n автомата f.

4) Обратная связь. Пусть дан линейный автомат $f(x_1,\ldots,x_n)$, и к переменной x_n может быть применена операция обратной связи, т.е. значение автомата f в момент времени t не зависит от $x_n(t)$. Замечу, что в терминах представления (4) к переменной x_n может быть применена операция обратной связи $\Leftrightarrow\ \mu_n(0) = 0$. Тогда [9]:

$$Fb_{x_n}\big(f(x_1,\ldots,x_n)\big) = g(x_1,\ldots,x_{n-1}) = \frac{1}{1+\mu_n}\left(\sum_{i=1}^{n-1}\mu_i x_i + \mu_0\right) =$$

$$= \sum_{i=1}^{n-1} \frac{\mu_i}{1+\mu_n} \cdot x_i + \frac{\mu_0}{1+\mu_n}. \qquad (11)$$

Таким образом, применение операции обратной связи к линейному автомату f по переменной x_n приводит к применению на коэффициентах автомата f операции Fb, определенной на элементах класса $E_2'(\xi)$ далее.

Операции композиции над линейными автоматами индуцируют следующие операции над элементами μ_1 и μ_2, $\mu_1, \mu_2 \in E_2^{'}(\xi)$:

1. Операция сложения элементов μ_1 и $\mu_2 : \ \mu_1 + \mu_2$.

2. Операция умножения элементов μ_1, $\mu_2 : \ \mu_1 \cdot \mu_2$.

3. Операция обратной связи, примененная к элементам μ_1 и μ_2 при условии, что $\mu_2(0) = 0 : Fb(\mu_1, \mu_2) = \frac{\mu_1}{1+\mu_2}$.

Также $E_2^{'}(\xi)$ можно считать множеством одноместных линейных автоматов, сохраняющих нулевую последовательность, к которым можно применять операции 1-3. Таким образом, на $E_2^{'}(\xi)$ вводится оператор $K^{(1)}$-замыкания. Аналогично введем и другие понятия:

M есть $K^{(1)}$-замкнутый класс $\Leftrightarrow \ M \subseteq E_2'(\xi), \ K^{(1)}(M) = M$.

M называется $K^{(1)}$-полным множеством $\Leftrightarrow M \subseteq E_2'(\xi), \ K^{(1)}(M) = E_2'(\xi)$.

Если $M \subset E_2'(\xi), \ M \neq E_2'(\xi), \ K^{(1)}(M) = M, \forall \ \mu \in E_2'(\xi) \backslash M \quad K^{(1)}(M \cup \{\mu\}) = E_2'(\xi)$, то M является $K^{(1)}$-предполным классом.

$K^{(1)}$-замкнутый класс M называется $K^{(1)}$-конечнопорожденным, если $\exists \ \{\mu_1, \ldots, \mu_m\} \subseteq M : M = K^{(1)}(\{\mu_1, \ldots, \mu_m\})$.

Пронумеруем неприводимые многочлены над E_2: $p_1 = \xi, \ p_2 = 1 + \xi, \ p_3 = 1 + \xi + \xi^2, \ldots$

В работе [9] были найдены все $K^{(1)}$-предполные классы, совокупность которых образует $K^{(1)}$-критериальную систему в $E_2'(\xi)$, т.е. $M \subseteq E_2'(\xi)$ будет $K^{(1)}$-полным множеством тогда, и только тогда, когда M не принадлежит ни одному классу этой системы.

В
$M_i^{(1)} = \big\{\mu \ \big| \ \mu \in E_2'(\xi), \ \mu + \mu(0) = \xi \cdot p_i \cdot \mu', \ \mu' = \frac{u'}{v'} \in E_2'(\xi), \ (v', p_i) = 1\big\}, \ i \in \mathbb{N}$.

Рассмотрим следующие подмножества автоматов в L_2.

$T_0 = \big\{f \big| \ f \in L_2, \ \mu_0(0) = 0\big\}$,

$T_1 = \big\{f \big| \ f \in L_2, \ \sum\limits_{i=0}^{n} \mu_i(0) = 1\big\}$,

$V_1 = \{f |\ f \in L_2,\ f$ имеет не более 1 непосредственной переменной$\}$,

$V_2 = \{f |\ f \in L_2,\ f$ имеет нечетное число непосредственных переменных$\}$,

$M_1 = \{f |\ f \in L_2, \forall\, \mu \in U(f), \mu \in M_1^{(1)}\}$.

Для данных множеств линейных автоматов имеет место следующая теорема:

Теорема [9].

$$M \subseteq L_2,\ A(M) = L_2 \Leftrightarrow M \not\subseteq \theta,\ \forall\, \theta \in \mathcal{J}_A = \{T_0, T_1, V_1, V_2, M_1\}. \quad (12)$$

То есть система классов $\mathcal{J}_A$ является A-критериальной системов в L_2.

3. K-конечнопорожденность предполных классов из системы предполных классов J_A

Лемма 1. *K-предполный класс линейных автоматов T_0 порождается множеством $M = \{\xi \cdot x,\ x_1 + x_2,\ \xi\}$, которое является базисом данного класса.*

Доказательство. Рассматривается класс автоматов $T_0 = \{f |\ f \in L_2,\ \mu_0(0) = 0\}$. Автомат этого класса f в представлении (4) имеет следующий вид:

$$f(x_1, \ldots, x_n) = \sum_{i=1}^{n} \mu_i x_i + \mu_0,\ \forall\, \mu_i \in E_2'(\xi),\ i = \overline{1,n},\ \mu_0 \in E_2'(\xi) : \mu_0(0) = 0. \quad (1)$$

Покажем, что $K(M) = T_0$. Обозначим элементы множества M: $f_1 = \xi \cdot x,\ f_2 = x_1 + x_2,\ f_3 = \xi$. Все три автомата принадлежат классу T_0, так как сохраняют ноль в начальный момент времени. Поскольку класс T_0 K-замкнут, то $K(M) \subset T_0$.

Нулевой автомат содержится в $K(M)$: $f_2(x, x) = x + x = 0 \in K(M)$.

Проводник принадлежит K-замыканию множества M: $f_2(x, 0) = x + 0 = x \in K(M)$.

Также верно, что $\forall\ m \in \mathbb{N}\ \ \xi^m x \in K(M)$, т.к. можно $(m-1)$ раз подставить автомат f_1 в себя: $f_1(f_1(..(f_1(x)\ldots) = \xi^m x \in K(M)$.

И для любого многочлена $u = \sum_{i=0}^{s} a_i \xi^i \in E_2[\xi],\ s \in \mathbb{Z}_+,\ u \cdot x \in K(M)$:

$$u \cdot x = \left(\sum_{i=0}^{s} a_i \xi^i \right) x = \sum_{i=0}^{s} a_i \xi^i x \in K(M). \quad (2)$$

Также можно показать, что для любого элемента $\mu \in E_2'(\xi)\ \ \mu \cdot x \in K(M)$. Действительно, произвольный элемент $E_2'(\xi)$ можно представить с помощью многочленов: $\mu = \dfrac{u}{v},\ u, v \in E_2[\xi],\ v(0) = 1,$ т.е. $v = 1+\xi v',\ v' \in E_2[\xi]$. Выше было доказано, что $\forall\ v' \in E_2[\xi]\ \ \xi \cdot v' \cdot x \in K(M)$, поэтому, применив операцию обратной связи к автомату $f(x_1, x_2) = u \cdot x_1 + \xi \cdot v' \cdot x_2 \in K(M)$ по переменной x_2 (данная переменная не является непосредственной, вследствие чего эта операция применима по x_2), получим искомый автомат:

$$Fb_{x_2}(f(x_1, x_2)) = \frac{u \cdot x_1}{1 + \xi \cdot v} = \mu \cdot x \in K(M). \tag{3}$$

Заметим, $f \in T_0 \Leftrightarrow \mu_0(0) = 0$. То есть свободный член автомата из класса T_0 можно представить так: $\mu_0 = \xi \cdot \mu',\ \mu' \in E_2'(\xi)$. Для любого $\mu' \in E_2'(\xi)\ \ f(x) = \mu' \cdot x \in K(M)$, поэтому, подставив в f автомат $f_3 \in M$, получим любую константу из T_0. Используя сумматор, можно получить любой автомат из T_0. Таким образом, $T_0 \subset K(M)$, благодаря чему $K(M) = T_0$.

Множество M является базисом класса T_0. Задержку исключить нельзя, т.к. в K-замыкании сумматора и ξ можно получить только функции вида $f(x_1, \ldots, x_n) = x_1 + + \ldots + x_n + a_0\xi,\quad a_0 \in E_2$, т.е. $\xi x \notin K(\{x_1 + x_2, \xi\})$. Без сумматора не получим автомата более, чем от одной переменной, что не есть весь класс $T_0 : K(M \backslash \{x_1 + x_2\}) \neq T_0$. Заметим, что сумматор и задержка сохраняют последовательность длины 2 $C = 00$, а ξ её не сохраняет (данный автомат выдает последовательность $\tilde{C} = 01$), поэтому ξ нельзя исключить из порождающей класс T_0 системы автоматов M. Можно показать, $\xi \notin K(\{f_1, f_2\})$. В связи с чем приходим к выводу, что множество M является базисом класса T_0. Лемма полностью доказана.

□

Лемма 2. *K-предполный класс линейных автоматов T_1 порождается множеством $M = \{f_1, f_2, f_3\} = \{x_1 + x_2 + x_3,\ \ \xi \cdot x + 1,\ 1\}$, которое является базисом данного класса.*

Доказательство. Класс T_1 есть класс автоматов, сохраняющих единицу в начальный момент времени. В представлении (4) это свойство можно выразить так:

$$f(x_1, \ldots, x_n) = \sum_{i=1}^{n} \mu_i x_i + \mu_0\ \in T_1 \Leftrightarrow \sum_{j=0}^{n} \mu_j(0) = 1. \tag{4}$$

Покажем, что $K(M) = T_1$. Каждый автомат из M сохраняет единицу в начальный момент, т.е. принадлежат классу T_1. В силу K-замкнутости класса T_1, $K(M) \subset T_1$.

Проводник можно получить в K-замыкании множества M, отождествив переменные автомата $f_1 : f_1(x, x, x) = x + x + x = x \in K(M)$.

Для любого натурального m получим автомат вида $(\xi^m + 1) \cdot x$. Для этого сначала подставим $(m-1)$ раз автомат f_2 в себя:

$$\tilde{f}_m(x) = \xi(\xi(\ldots\xi(\xi x + 1) + 1)\ldots) + 1 =$$

$$= \xi^m \cdot x + (\xi^{m-1} + \xi^{m-2} + \ldots + \xi + 1) \in K(M). \tag{5}$$

Далее, подставляя f_3 в автоматы $\tilde{f}_m(x)$, можно получить для любого m все константы вида $c_m = (\xi^m + \xi^{m-1} + \xi^{m-2} + \ldots + \xi + 1) \in K(M)$. Тогда сделаем следующее:

$$f_1(\tilde{f}_m(x), c_{m-1}, x) = \big(\xi^m \cdot x + (\xi^{m-1} + \xi^{m-2} + \ldots + \xi + 1)\big) +$$

$$+ (\xi^{m-1} + \xi^{m-2} + \ldots + \xi + 1) + x = (\xi^m + 1) \cdot x \in K(M). \tag{6}$$

Покажем, что для любого многочлена $\tilde{u} \in E_2[\xi]$ в K-замыкании множества M есть автомат $(1 + \xi\tilde{u}) \cdot x \in K(M)$. Пусть $\tilde{u} = a_1 + a_2\xi + \ldots + a_s\xi^{s-1} + a_{s+1}\xi^s$, $s \in \mathbb{N}$. Также введем многочлен u:

$$u = 1 + \xi\tilde{u} = 1 + \xi(a_1 + a_2\xi + \ldots + a_s\xi^{s-1} + a_{s+1}\xi^s) =$$

$$= 1 + a_1\xi + a_2\xi^2 + \ldots + a_s\xi^s + a_{s+1}\xi^{s+1}. \tag{7}$$

Как было показано выше, для любого натурального m, в том числе и для $m : m = \overline{1, (s+1)}$, $g_m(x) = (a_m\xi^m + 1) \cdot x \in K(M)$ (если $a_m = 0$, то соответствующий член многочлена является проводником, который также есть в $K(M)$).

Рассмотрим 2 случая:

1) s - четное число. Тогда слагаемых в $\tilde{u}$ $(s+1)$ нечетное количество. В таком случае, в сумматор от $s+1$ переменной f_{s+1} (подставляя автомат f_1 в себя, в $K(M)$ можно получить сумматор от любого нечетного числа переменных) подставляются автоматы $g_m(x)$, $m = \overline{1, (s+1)}$:

$$f_{s+1}(g_1(x), \ldots, g_{s+1}(x)) = \sum_{m=1}^{s+1}(a_m\xi^m + 1) \cdot x = \sum_{m=1}^{s+1} a_m\xi^m \cdot x + x =$$

$$= (1 + a_1\xi + \ldots + a_{s+1}\xi^{s+1})x = (1 + \xi(a_1 + \ldots + a_{s+1}\xi^s))x = (1 + \xi\tilde{u})x \in K(M). \tag{8}$$

2) s - нечетное число, т.е. в $\tilde{u}$ $(s+1)$ слагаемых четное количество. Тогда в сумматор от $s+2$ переменных f_{s+2} подставляются автоматы $g_m(x),\ m=\overline{1,(s+1)}$ и проводник:

$$f_{s+2}(g_1(x),\dots,g_{s+1}(x),x)=\sum_{m=1}^{s+1}(a_m\xi^m+1)\cdot x+x=$$

$$=\sum_{m=1}^{s+1}a_m\xi^m\cdot x+x=(1+\xi\tilde{u})x\in K(M). \tag{9}$$

Также для любого $\mu\in E_2'(\xi)$ верно, что $(1+\xi\mu)\cdot x\in K(M)$. Представим μ в виде формальной дроби:

$$\mu=\frac{u}{1+\xi\cdot v},\ \ u,v\in E_2[\xi]. \tag{10}$$

Знаем, что $\forall\ u',v\in E_2[\xi]\quad (1+\xi u')x,\ (1+\xi v)x\in K(M)$ $(u=u'+v,u'=u+v)$. Подставим эти автоматы в f_1 и отождествим две переменные:

$$f_1((1+\xi u')x,(1+\xi v)x_1,x_1)=(1+\xi u')x+(1+\xi v)x_1+x_1=$$

$$=(1+\xi u')x+\xi v x_1\in K(M). \tag{11}$$

Затем применим операцию обратной связи по x_1 (переменная x_1 не является непосредственной, поэтому операция обратной связи применима по данной переменной):

$$Fb_{x_1}(f_1((1+\xi u')x,(1+\xi v)x_1,x_1))=\frac{1+\xi u'}{1+\xi v}x=$$

$$=\left(1+\xi\cdot\frac{u'+v}{1+\xi v}\right)x=(1+\xi\mu)x\in K(M). \tag{12}$$

Подставив в $(1+\xi\mu)x$ автомат f_3, получим в $K(M)$ любую константу вида $(1+\xi\mu)$.

Введем ещё 2 вспомогательные функции из $K(M)$:

$$h_1(x,x_1)=f_1((1+\xi\mu)x,x,x_1)=(1+\xi\mu)x+x+x_1=\xi\mu x+x_1\in K(M) \tag{13}$$

$$h_2(x_1)=h_1(1,x_1)=\xi\mu+x_1\in K(M) \tag{14}$$

Для автомата из T_1 верно, что среди $n+1$ слагаемых его разложения (n переменных и 1 константа) нечетное число s слагаемых имеет вид (1): или $(1+\xi\mu)x$, или $(1+\xi\mu)$; остальные m слагаемых имеют вид (2): $\xi\mu x$ или $\xi\mu$.

Можно выделить 4 случая построения произвольного автомата из T_1.

1) n — четное число и свободный член автомата имеет вид (1). Тогда $s-1$ переменная имеет вид (1), оставшиеся $n-(s-1)=m$ (четное число) переменных имеют вид (2). В сумматор от $(n+1)$ переменной подставляются s автоматов вида (1) (они есть в $K(M)$), в оставшиеся позиции подставляются автоматы $h_1(x,x_1)$, при этом переменная x_1 отождествляется и, ввиду четности m, сокращается.

2) n — четное число и свободный член автомата имеет вид (2). Тогда s переменных имеет вид (1), оставшиеся $n-s=m-1$ (нечетное число) переменных имеют вид (2). В сумматор от $(n+1)$ переменной подставляются s автоматов вида (1) (они есть в $K(M)$), в оставшиеся позиции подставляются автоматы $h_1(x,x_1),\ h_2(x_1)$, при этом в полученном автомате переменная x_1, ввиду четности m, сокращается.

3) n — нечетное число и свободный член автомата имеет вид (1). Тогда $s-1$ переменных имеет вид (1), оставшиеся $n-(s-1)=m$ (нечетное число) переменных имеют вид (2). В сумматор от $(n+2)$ переменной подставляются s автоматов вида (1) (они есть в $K(M)$), в оставшиеся позиции подставляются автоматы $h_1(x,x_1)$ и автомат $g(x_1)=x_1$, при этом переменная x_1 отождествляется, и, ввиду четности $(m+1)$, сокращается.

4) n — нечетное число и свободный член автомата имеет вид (2). Тогда s переменных имеет вид (1), оставшиеся $n-s=m-1$ (четное число) переменных имеют вид (2). В сумматор от $(n+2)$ переменной подставляются s автоматов вида (1) (они есть в $K(M)$), в оставшиеся позиции подставляются автоматы $h_1(x,x_1),\ h_2(x_1)$ и автомат $g(x_1)=x_1$, при этом, ввиду четности m, переменная x_1 сокращается.

Подобным образом может быть получен любой автомат из класса T_1. Следовательно, $T_1 \subset K(M)$. Что приводит нас к заключению, что $T_1 = K(M)$.

Покажем, что множество M является базисом класса T_1. Сумматор $x_1+x_2+x_3$ является единственным автоматом в M от более, чем от одной переменной, поэтому $K(\{\xi\cdot x+1,\ 1\}) \neq T_1$. Без автомата $f_2=\xi\cdot x+1$ с помощью операций композиции из множества M можно получить только автоматы вида $f(x_1,\ldots,x_n)=x_1+\ldots+x_n+a_0\xi,\ a_0\in E_2$, т.е. $K(\{f_1,f_3\})\neq T_1$. Заметим, что сумматор от трех переменных сохраняет любую последовательность произвольной длины. Автомат f_3 не содержится в K-замыкании $\{x_1+x_2+x_3,\ \xi\cdot x+1\}$, т.к. автоматы f_1, f_2 сохраняют последовательность длины 2: $C=11$, а f_3 выдает последовательность 10. Можно показать, что автомата f_3 в K-замыкании множества $\{f_1,f_2\}$ нет. В связи с чем, множество M является базисом класса T_1. Лемма доказана полностью.

□

Лемма 3. *K-предполный класс линейных автоматов V_1 порождается множеством $M = \{f_1, f_2, f_3\} = \{\xi x_1 + x_2,\ \xi \cdot x,\ x+1\}$, которое является базисом данного класса.*

Доказательство. Автомат класса $V_1 = \{f \mid f \in L_2,\ f$ имеет не более 1 непосредственной переменной $\}$ в разложении (4) представляется в таком виде: $f(x_1, \ldots, x_n) = \sum\limits_{i=1}^{n} \mu_i x_i + \mu_0 \ \in V_1$.

x_i — непосредственная переменная автомата $f \Leftrightarrow \mu_i(0) = 1 \Leftrightarrow \mu_i = 1 + \xi\mu',\ \mu' \in E_2'(\xi)$.

Докажем, что $K(M) = V_1$. Каждый автомат множества M имеет не более 1 непосредственной переменной, и, поскольку класс V_1 является K-замкнутым, $K(M) \subset V_1$.

Нулевой автомат содержится в $K(M) : Fb_x(\xi x) = 0 \in K(M)$.

Проводник также есть в $K(M) : f_1(0, x) = x = 1 \cdot x \in K(M)$.

Подставив автомат f_2 в себя $m-1$ раз (для любого натурального m), получим автомат $\xi^m x \in K(M)$:

$$f_2(f_2(..(f_2(x)\ldots) = \xi^m x \in K(M). \tag{15}$$

Покажем, что для любого многочлена $u = \sum\limits_{i=0}^{s} = a_i\xi^i \in E_2[\xi],\ s \in \mathbb{Z}_+,\ u \cdot x \in K(M)$. Докажем этот факт для произвольного многочлена u с помощью математической индукции по степени многочлена s. Действительно, многочлены нулевой степени ($u = 0$, $u = 1$) реализуется в $K(M)$ с помощью нулевого автомата и проводника, присутствующих в $K(M)$. Для $s = 1$ многочлену $u = \xi$ соответствует автомат f_2, а для многочлена $u = \xi + 1$ в замыкании $K(M)$ есть такой автомат: $f_1(x, x) = \xi \cdot x + x = (\xi + 1)x \in K(M)$.

Пусть данный факт доказан для многочленов степени s. Тогда для любого многочлена степени $(s+1)$ $u = a_0 + \ldots + a_s\xi^s + \xi^{s+1}$ получим искомый автомат следующим образом:

$$f_1(\xi^s x, (a_0 + \ldots + a_s\xi^s)x) = \xi(\xi^s x) + (a_0 + \ldots + a_s\xi^s)x =$$

$$= (a_0 + \ldots + a_s\xi^s + \xi^{s+1})x = u \cdot x \in K(M). \tag{16}$$

Также для любого $\mu \in E_2'(\xi)$ верно, что $\mu \cdot x \in K(M)$. Представим элемент $E_2'(\xi)$ в виде дроби многочленов:

$$\mu = \frac{u}{1 + \xi \cdot v} \in E_2'(\xi),\ u, v \in E_2[\xi]. \tag{17}$$

В автомат f_1 подставим автоматы $ux, vx \in K(M) : f_1(vx_1, ux_2) = \xi vx_1 + ux_2 \in K(M)$. И применим операцию обратной связи по не непосредственной переменной x_1:

$$Fb_{x_1}(f_1(vx_1, ux_2)) = \frac{u}{1+\xi v} \cdot x_2 = \mu \cdot x_2 \in K(M). \tag{18}$$

Подставим в автомат f_3 нулевой автомат: $f_3(0) = 1 \in K(M)$. Подставив, в свою очередь, единицу в автоматы $\mu \cdot x$, получим в K-замыкании множества M любую константу μ из $E_2'(\xi)$.

Так как любой элемент $E_2'(\xi)$ представим или в виде $\mu = \xi\mu'$, или в виде $\mu = 1 + \xi\mu'$, $\mu' \in E_2'(\xi)$, то автомат $x + \mu$ также можно получить в K-замыкании множества M.
В первом случае в автомат f_1 подставим константу $\mu' \in K(M)$:

$$f_1(\mu', x) = \xi\mu' + x \in K(M). \tag{19}$$

Во втором случае в автомат f_3 поставим только что полученный автомат $\xi\mu' + x$:

$$f_3(\xi\mu' + x) = \xi\mu' + x + 1 = x + (1 + \xi\mu') = x + \mu \in K(M). \tag{20}$$

Заметим,что любой автомат f из V_1 имеет следующий вид:

$$f(x_1, \dots x_n) = \sum_{i=1}^{n-1} \xi\mu_i' x_i + \mu_n x_n + \mu_0, \text{ где } \mu_n, \mu_0, \mu_i' \text{ любые элементы } E_2'(\xi). \tag{21}$$

Покажем, что любой такой автомат есть в $K(M)$. Подставляя на второй вход автомата f_1 самого себя, получим: $f_1(x_1, f_1(x_2, \dots f_1(x_{n-1}, x_n) \dots) = = \sum_{i=1}^{n-1} \xi x_i + x_n$. Далее поставим на первые $n-1$ входов автоматы $\mu_i' x_i \in K(M)$, а на n-ый вход подставляется автомат $\mu_n x_n + \mu_0 \in K(M)$. Таким образом, любой автомат из класса V_1 содержится в K-замыкании множества $M : V_1 \subset K(M)$. И в связи с вышеизложенным получаем, что $V_1 = K(M)$.

Множество M является базисом класса V_1. Автоматы f_1 и f_3 сохраняют в начальный момент функции $\{x, \bar{x}\}$, а автомат f_2 не сохраняет (в начальный момент у данного автомата тождественный ноль). Можно показать, что ввиду наличия у f_2 такого свойства $f_2 \notin K(\{f_1, f_3\})$. Так как автомат f_1 является единственным автоматом от более, чем одной переменной, то $K(\{f_2, f_3\}) \neq V_1$. Заметим, что автоматы f_1, f_2 также принадлежат классу автоматов T_0, а автомат f_3 нет. В силу K-замкнутости класса $T_0 : f_3 \notin K(\{f_1, f_2\})$. В результате приходим к выводу, что множество M — базис класса V_1. Лемма полностью доказана.

□

Лемма 4. *K-предполный класс линейных автоматов V_2 порождается множеством $M = \{f_1, f_2, f_3\} = \{x_1 + x_2 + x_3,\ \xi \cdot x_1 + x_2,\ x + 1\}$, которое является базисом данного класса.*

Доказательство. Рассматриваемый класс V_2 — это класс линейных автоматов, имеющих нечетное число непосредственных переменных. В разложении (4) это свойство можно выразить следующим образом:

$$f(x_1, \ldots, x_n) = \sum_{i=1}^{n} \mu_i x_i + \mu_0 \ \in V_2 \Leftrightarrow \sum_{i=1}^{n} \mu_i(0) = 1. \tag{22}$$

Покажем, что $K(M) = V_2$. Автомат f_1 имеет 3 непосредственные переменные, автоматы f_2, f_3 — 1 непосредственную переменную. Так как класс V_2 K-замкнут, то $K(M) \subset V_2$.

Отождествив переменные автомата f_1, получим в K-замыкании множества M проводник: $f_1(x, x, x) = x + x + x = x \in K(M)$.

Также в $K(M)$ для любого натурального m есть автомат $(\xi^m + 1)x$. Подставим $m - 1$ раз автомат f_2 в себя и отождествим переменные результирующего автомата:

$$\hat{f}_m(x) = \xi(\xi(\ldots\xi(\xi x + x) + x)\ldots + x) + x = (\xi^m x + \xi^{m-1} x + \xi^{m-2} x + \ldots + \xi x + x) =$$

$$= (\xi^m + \xi^{m-1} + \xi^{m-2} + \ldots + \xi + 1)x \in K(M). \tag{23}$$

Затем подставим две подобные автоматные функции в f_1 и аналогично отождествим все переменные:

$$\tilde{f}_m(x) = f_1(\hat{f}_m(x), \hat{f}_{m-1}(x), x) = (\xi^m + \xi^{m-1} + \xi^{m-2} + \ldots + \xi + 1)x +$$

$$+ (\xi^{m-1} + \xi^{m-2} + \ldots + \xi + 1)x + x = (\xi^m + 1)x \in K(M). \tag{24}$$

Для любого многочлена $\tilde{u} \in E_2[\xi]$ в $K(M)$ можно получить автомат вида $(1 + \xi\tilde{u})x$. Обозначим $\tilde{u} = a_1 + a_2\xi + \ldots + a_s\xi^{s-1} + a_{s+1}\xi^s,\ s \in \mathbb{Z}_+$. Также введем многочлен u:

$$u = 1 + \xi\tilde{u} = 1 + \xi(a_1 + a_2\xi + \ldots + a_s\xi^{s-1} + a_{s+1}\xi^s) =$$

$$= 1 + a_1\xi + a_2\xi^2 + \ldots + a_s\xi^s + a_{s+1}\xi^{s+1}. \tag{25}$$

Докажем это включение, используя метод математической индукции по степени многочлена s. Для многочленов нулевой степени:

$$\tilde{u} = 0 \Rightarrow (1 + 0 \cdot \xi)x = x \in K(M); \tag{26}$$

$$\tilde{u} = 1 \Rightarrow (1 + 1 \cdot \xi)x = f_2(x, x) = \xi x + x \in K(M). \tag{27}$$

Для многочленов первой степени верно: $\tilde{u} = a_1 + \xi \Rightarrow u = 1 + \xi(a_1 + \xi) = 1 + a_1\xi + \xi^2$. Подставим в автомат f_1 автоматы $\tilde{f}_2(x),\ (1 + a_1\xi)x \in K(M)$:

$$f_1(\tilde{f}_2(x), (1 + a_1\xi)x, x) = (\xi^2 + 1)x + (1 + a_1\xi)x + x =$$

$$= (1 + a_1\xi + \xi^2)x = (1 + \xi(a_1 + \xi))x \in K(M). \tag{28}$$

Пусть для многочленов $\tilde{u}$ степени s утверждение верно. Докажем его для любого многочлена $\tilde{u}$ степени $s+1$. В автомат f_1 подставим автомат $\tilde{f}_{s+2}(x) \in K(M)$ и автомат $(1 + a_1\xi + + \ldots + a_{s+1}\xi^{s+1})x = (1 + \xi(a_1 + \ldots + a_{s+1}\xi^s))x \in K(M)$ (по предположению индукции такой автомат в K-замыкании множества M есть):

$$f_1(\tilde{f}_{s+2}(x),\ (1\ +\ a_1\xi\ +\ \ldots\ +\ a_{s+1}\xi^{s+1})x,\ x) = (\xi^{s+2}\ +\ 1)x\ +\ (1 +$$

$$+\ a_1\xi\ +\ \ldots\ +\ a_{s+1}\xi^{s+1})x\ +\ x = (1 + a_1\xi + \ldots + a_{s+1}\xi^{s+1} + \xi^{s+2})x =$$

$$= (1 + \xi(a_1 + \ldots + a_{s+1}\xi^s + \xi^{s+1}))x \in K(M). \tag{29}$$

Для любого $\mu \in E_2'(\xi)$ покажем, что $(1 + \xi\mu) \cdot x \in K(M)$. В виде формальной дроби μ представляется так:

$$\mu = \frac{u}{1 + \xi \cdot v},\ \ u, v \in E_2[\xi]. \tag{30}$$

Подставим в сумматор от трех переменных автоматы $(1 + \xi u')x,\ (1 + +\xi v)x$ и $x \in K(M)\ \ \forall\, u', v \in E_2[\xi]\ \ (u = u' + v, u' = u + v)$ соответственно и отождествим вторую и третью переменные:

$$g(x_1, x_2) = f_1((1 + \xi u')x_1, (1 + \xi v)x_2, x_2) =$$

$$= (1 + \xi u')x_1 + (1 + \xi v)x_2 + x_2 = (1 + \xi u')x + \xi v x_2 \in K(M). \tag{31}$$

Затем применяется операция обратной связи по не непосредственной переменной x_2:

$$Fb_{x_2}(g(x_1, x_2)) = \frac{1 + \xi u'}{1 + \xi v}x_1 = \left(1 + \xi \cdot \frac{u' + v}{1 + \xi v}\right)x_1 = (1 + \xi\mu)x_1 \in K(M). \tag{32}$$

Далее покажем, что для любой константы μ' из $E_2'(\xi)$ автомат $\mu x + +\mu' \in K(M)$ $(\mu = 1 + \xi\hat{\mu},\ \hat{\mu} \in E_2'(\xi))$. Для этого сначала докажем это для любого $\mu' = \tilde{u}$, где $\tilde{u} \in E_2[\xi]$. Многочлены нулевой степени: $\mu x + 0 \in K(M),\quad \mu x + 1 = f_3(\mu x) \in K(M)$. Для многочленов первой степени также рассмотрим два случая.

Первый случай: $\tilde{u} = \xi$. Здесь на первый вход автомата f_2 подставим автомат f_3, затем отождествим переменные и получим следующее:

$$f_2(f_3(x), x) = \xi(x+1) + x = (1+\xi)x + \xi \in K(M). \tag{33}$$

Искомый автомат получим с помощью автомата f_1:

$$f_1((1+\xi)x + \xi, (1+\xi)x, \mu x) = \mu x + \xi \in K(M) \tag{34}$$

Второй случай: $\tilde{u} = \xi + 1$. В автомат f_3 подставим автомат из случая 1:

$$f_3(\mu x + \xi) = \mu x + \xi + 1 \in K(M). \tag{35}$$

Заметим, что для любого натурального m автомат $\xi^m + \mu x \in K(M)$. Действительно, пусть это утверждение верно для $m-1$ (для $m = 1$ доказано выше). Тогда:

$$f_2(\xi^{m-1} + x, x) = \xi(\xi^{m-1} + x) + x = \xi^m + (1+\xi)x \in K(M) \tag{36}$$

$$\tilde{g}_m(x) = f_1(f_2(\xi^{m-1} + x, x), (1+\xi)x, \mu x) =$$

$$= \xi^m + (1+\xi)x + (1+\xi)x + \mu x = \xi^m + \mu x \in K(M). \tag{37}$$

Пусть для любого многочлена $\tilde{u}$ степени $m-1$ доказано, что $\mu x + \tilde{u} \in K(M)$. Покажем, что для любого многочлена $u = a_0 + a_1 \cdot \xi + \ldots + a_{m-1} \cdot \xi^{m-1} + \xi^m$ это утверждение также верно. Для этого в автомат f_1 подставим автоматы из K-замыкания множества M $(\xi^m + \mu x)$, μx, $(a_0 + +a_1\xi + \ldots + a_{m-1}\xi^{m-1}) + \mu x$:

$$f_1\Big(\big((a_0 + a_1\xi + \ldots + a_{m-1}\xi^{m-1}) + \mu x\big), \big(\xi^m + \mu x\big), \mu x\Big) = (a_0 +$$

$$+a_1\xi + \ldots + a_{m-1}\xi^{m-1}) + \mu x + \big(\xi^m + \mu x\big) + \mu x = (a_0 +$$

$$+ a_1\xi + \ldots + a_{m-1}\xi^{m-1} + \xi^m) + \mu x = \mu x + \tilde{u} \in K(M). \tag{38}$$

Также можно получить, что $\forall\ \mu_0 = \dfrac{u}{1+\xi v} \in E_2'(\xi),\ \ u, v \in E_2[\xi]\ \ \mu_0 + \mu x \in K(M)\ \ (\mu : \mu(0) = 1)$. Сначала введем вспомогательный автомат:

$$g(x_1, x_2) = f_1\big(u + (1+\xi v)\mu x_1,\ (1+\xi v)x_2,\ x_2\big) = u + (1+\xi v)\mu x_1 +$$

$$+ (1+\xi v)x_2 + x_2 = u + (1+\xi v)\mu x_1 + \xi v x_2 \in K(M). \tag{39}$$

А затем применим операцию обратной связи по не непосредственной переменной x_2:

$$Fb_{x_2}(g(x_1,x_2)) = \frac{u}{1+\xi v} + \frac{1+\xi v}{1+\xi v}\mu x_1 = \mu_0 + \mu x_1 \in K(M). \qquad (40)$$

И, наконец, докажем, что любой автомат из V_2 от любого количества переменных принадлежит $K(M)$. Действительно, все автоматы из V_2 от одной переменной в $K(M)$ есть. Автоматы V_2 от двух переменных имеют одну непосредственную переменную и одну не непосредственную:

$$f(x_1,x_2) = (1+\xi\mu_1')x_1 + \xi\mu_2' x_2 + \mu_0 \in V_2 \qquad (41)$$

Таким автоматы содержатся в $K(M)$:

$$f_1((1+\xi\mu_1')x_1 + \mu_0,\ (1+\xi\mu_2')x_2,\ x_2) = f(x_1,x_2) \in V_2 \qquad (42)$$

Пусть любой автомат $\hat{f}(x_1,\ldots,x_n)$ из V_2 от n переменных принадлежит $K(M)$. Чтобы увеличить количество непосредственных переменных для сохранности нечетности их числа, необходимо добавить сразу 2 непосредственные переменные x_{n+1} и x_{n+2} :

$$\tilde{f}(x_1,\ \ldots,\ x_n,\ x_{n+1},\ x_{n+2}) = \hat{f}(x_1,\ \ldots,\ x_n)\ +\ (1\ +\ \xi\mu_{n+1}')x_{n+1} +$$
$$+\ (1\ +\ \xi\mu_{n+2}')x_{n+2} \in K(M). \qquad (43)$$

Чтобы увеличить количество не непосредственных переменных достаточно добавить одну x_{n+1} переменную:

$$\tilde{f}(x_1,\ldots,x_n,x_{n+1}) = \hat{f}(x_1,\ldots,x_n) + (1+\xi\mu_{n+1})\ x_{n+1} +\ x_{n+1} =$$
$$= \hat{f}(x_1,\ldots,x_n) + \xi\mu_{n+1}\ x_{n+1} \in K(M). \qquad (44)$$

Таким образом, любой автомат из V_2 содержится в $K(M)$: $V_2 \subset K(M)$. Следовательно, с учетом вышеизложенного $K(M) = V_2$.

Докажем, что порождающая система M является базисом класса V_2. Действительно, автоматы $f_1, f_2 \in T_0$ — K-замкнутому классу, отличному от V_2, т.е. $K(\{f_1,f_2\}) \neq V_2$ (автомат $f_3 \notin T_0$). Исключить автомат f_2 не получится, поскольку в K-замыкании сумматора от трех переменных и автомата f_3 можно получить только автоматы вида $f(x_1,\ldots,x_{2m+1}) =$

$x_1 + \ldots + x_{2m+1} + a_0$, $a_0 \in E_2$, т.е. $f_2 \notin K(\{f_1, f_3\})$. А без сумматора от трех переменных нельзя получить автомат, у которого будет n непосредственных переменных для любого нечетного числа n. Заметим, что среди операций композиции только операция подстановки позволяет увеличивать количество переменных автомата, но с помощью этой операции, в K-замыкании автоматов $f_2,\ f_3$ нельзя получить автомат более, чем от одной непосредственной переменной. На основании вышеизложенного приходим к выводу, что множество M является базисом класса V_2. Лемма доказана полностью.

□

Лемма 5. *K-предполный класс линейных автоматов M_1 порождается множеством $M = \{f_1, f_2, f_3, f_4\} = \{\xi^3 x_1 + x_2 + x_3,\ \xi^2 \cdot x,\ 1,\ \xi\}$, которое является базисом данного класса.*

Доказательство. По определению класса M_1 любой коэффициент $\mu \in U(f)$ при переменной принадлежит классу $M_1^{(1)}$. В разложении (4) любой автомат класса M_1представляется так:

$$f(x_1, \ldots, x_n) = \sum_{i=1} \mu_i x_i + \mu_0 \ \in M_1 \Leftrightarrow \mu_i \in M_1^{(1)},\ i = \overline{1,n},\ \mu_0 \in E_2'(\xi). \tag{45}$$

В свою очередь, свойство принадлежности $\mu_i \in M_1^{(1)}$ можно выразить следующим образом: $\mu_i \in M_1^{(1)} \Leftrightarrow \mu_i = a_0 + \xi^2 \mu_i',\ a_0 \in E_2,\ \forall\ \mu_i' \in E_2'(\xi)$. Другими словами, автомат класса M_1 выглядит так:

$$f(x_1, \ldots, x_n) = \sum_{i=1}^{n} (a_0^i + \xi^2 \mu_i') x_i + \mu_0 \ \in M_1,\ a_0 \in E_2,\ \ \mu_i',\ \mu_0 \in E_2'(\xi). \tag{46}$$

Докажем, что $K(M) = M_1$. Автоматы f_1, f_2 содержатся в классе M_1, т.к. их коэффициенты при переменных принадлежат классу $M_1^{(1)}$. Автоматы f_3, f_4 — это константы, а классу M_1 принадлежат все константы из $E_2'(\xi)$. Поэтому, в силу K-замкнутости класса M_1, $K(M) \subset M_1$.

В $K(M)$ есть нулевой автомат: $Fb_x(f_2(x)) = 0 \in K(M)$.

Проводник также принадлежит K-замыканию множества M: $f_1(0, 0, x) = x \in K(M)$.

Автомат $\xi^3 x$ аналогично выводится из автомата f_1 подстановкой нулевого автомата на его второй и третий входы: $f_1(x, 0, 0) = \xi^3 x \in K(M)$.

Заметим, что подставив на первый вход автомата f_1 нулевой автомат, мы получим сумматор от двух переменных: $f_1(0, x_2, x_3) = x_2 + x_3 \in K(M)$. То есть достаточно показать, что для любого $\mu \in E_2'(\xi)$ элементы вида $(a_0 + \xi^2 \mu)x$ и μ принадлежат K-замыканию множества M.

Докажем, что для любого многочлена $\tilde{u} \in E_2[\xi]$ в $K(M)$ можно получить автомат вида $\xi^2 x \tilde{u} \in K(M)$, используя математическую индукцию

по степени многочлена s. Действительно, для $s = 0$ было показано,что такие автоматы принадлежат K-замыканию множества M:

$$\xi^2 x \cdot 0 = 0 \in K(M); \tag{47}$$

$$\xi^2 x \cdot 1 = \xi^2 x = f_2(x) \in M. \tag{48}$$

Многочлены первой степени:

$$\xi^2 x \cdot \xi = \xi^3 x \in K(M); \tag{49}$$

$$\xi^2 x \cdot (1 + \xi) = f_1(x, \xi^2 x, 0) = \xi^3 x + \xi^2 x = \xi^2 x(1 + \xi) \in K(M). \tag{50}$$

Пусть для любого многочлена $\tilde{u} = a_0 + a_1\xi + \ldots + a_{s-1}\xi^{s-1}$ степени $s-1$ утверждение верно. Докажем,что соответствующий автомат содержится в $K(M)$ и для любого многочлена степени s. Для этого достаточно показать, что для любого натурального $s \in \mathbb{N}$ автомат $\xi^2 \cdot \xi^s x$ принадлежит K-замыканию множества M (для $s = 1$ доказано выше). Рассмотрим 2 случая:

1) $s = 2 \cdot m,\ m \in \mathbb{N}$. Для этого случая достаточно m раз поставить автомат f_2 в себя:

$$f_2(f_2(\ldots f_2(f_2(x))\ldots) = \xi^2(\xi^2(\ldots\xi^2(\xi^2 x)\ldots) =$$

$$= \xi^2 \cdot \xi^{2\cdot m} x = \xi^2 \cdot \xi^s x \in K(M). \tag{51}$$

2) $s = 2\cdot m+1,\ \ m \in \mathbb{N}$. Искомый автомат можно получить, подставив в автомат $\xi^3 x\ \ m$ раз автомат f_2:

$$\xi^3(f_2(\ldots f_2(f_2(x))\ldots) = \xi^3(\xi^2(\xi^2(\ldots(\xi^2 x)\ldots) =$$

$$= \xi^{2+2(m-1)+3} x = \xi^2 \cdot \xi^s x \in K(M). \tag{52}$$

Очевидно, $2(m-1)+3 = 2\cdot m+1$.

Таким образом, для любого многочлена u степени s (обозначим $u = \tilde{u} + \xi^s$) автомат $\xi^2 x \cdot u$ в $K(M)$ есть:

$$\xi^2 \tilde{u} x + \xi^2 \cdot \xi^s x = \xi^2(\tilde{u} + \xi^s)x = \xi^2 \cdot ux \in K(M). \tag{53}$$

Покажем, что для любого элемента $\mu \in E_2'(\xi)$ автомат $\xi^2 \mu x$ принадлежит K-замыканию множества M. Произвольный элемент из $E_2'(\xi)$ можно представить так:

$$\mu = \frac{u}{1 + \xi \cdot v},\ \ u, v \in E_2[\xi]. \tag{54}$$

Также рассмотрим 2 случая:

1) $v = 0 + a_1\xi + a_2\xi^2 + \ldots + \xi^s = \xi(a_1 + a_2\xi + \ldots + \xi^{s-1}) = \xi\tilde{v}$. То есть μ имеет следующий вид:

$$\mu = \frac{u}{1 + \xi^2 \cdot \tilde{v}}, \quad u, \tilde{v} \in E_2[\xi]. \tag{55}$$

Тогда сначала подставим в сумматор от двух переменных автоматы $\xi^2 ux$, $\xi^2\tilde{v}x \in K(M)$: $\tilde{f}(x_1, x_2) = \xi^2 ux_1 + \xi^2\tilde{v}x_2 \in K(M)$. После чего применим операцию обратной связи по второй переменной:

$$Fb_{x_2}(\tilde{f}(x_1, x_2)) = \frac{\xi^2 u}{1 + \xi^2\tilde{v}}x_1 = \xi^2\frac{u}{1 + \xi^2\tilde{v}}x_1 = \xi^2\mu x \in K(M). \tag{56}$$

2) $v = 1 + a_1\xi + a_2\xi^2 + \ldots + \xi^s = 1 + \xi\tilde{v}$. Соответствующий элемент $E_2'(\xi)$ выглядит так:

$$\mu = \frac{u}{1 + \xi + \xi^2 \cdot \tilde{v}}, \quad u, \tilde{v} \in E_2[\xi]. \tag{57}$$

Покажем, что вспомогательный автомат $g(x) = \dfrac{1}{1 + \xi^2}x$ содержится в K-замыкании M. Действительно, подставим автомат f_2 на первый вход сумматора от двух переменных и применим по этой переменной операцию обратной связи:

$$g(x) = Fb_{x_1}(\xi^2 x_1 + x_2) = \frac{1}{1 + \xi^2}x_2 \in K(M). \tag{58}$$

Заметим, что для поля E_2 верно $(1+\xi^2) = (1+\xi)^2$. Для рассматриваемого случая сначала подставим на оба входа сумматора от двух переменных автомат $g(x)$:

$$\hat{g}(x_1, x_2) = \frac{1}{1 + \xi^2}x_1 + \frac{1}{1 + \xi^2}x_2 \in K(M). \tag{59}$$

Затем, поскольку для любого многочлена u из кольца $E_2'[\xi]$ доказано, что $\xi^2 ux$ есть в K-замыкании M, то автоматы $\xi^2(1+\xi)\tilde{v}x$ и $\xi^2(1+\xi)ux$ также содержатся в $K(M)$. Подставим их на входы только что полученного автомата $\hat{g}(x_1,\ x_2)$:

$$\tilde{g}(x_1,\ x_2) \ = \ \hat{g}(\xi^2(1\ +\ \xi)\tilde{v}x_1,\ \xi^2(1\ +\ \xi)ux_2) \ = \ \frac{1}{1\ +\ \xi^2}\Big(\xi^2(1\ +\ \xi)\tilde{v}\Big)x_1 \ +$$

$$+\frac{1}{1+\xi^2}\Big(\xi^2(1+\xi)u\Big)x_2=\frac{\xi^2\tilde{v}}{1+\xi}x_1+\frac{\xi^2u}{1+\xi}x_2\in K(M). \qquad (60)$$

И, наконец, применим операцию обратной связи по первой переменной:

$$Fb_{x_1}(\tilde{g}(x_1,x_2))=\frac{\xi^2u}{1+\xi}\cdot\frac{1}{1+\frac{\xi^2\tilde{v}}{1+\xi}}x_2=\frac{\xi^2u}{1+\xi}\cdot\frac{1+\xi}{1+\xi+\xi^2\tilde{v}}x_2=$$

$$=\frac{\xi^2u}{1+\xi+\xi^2\tilde{v}}x_2=\xi^2\mu x\in K(M). \qquad (61)$$

Таким образом, для любого $\mu=\xi^2\mu',\ \mu'\in E_2'(\xi)$ доказано, что $\mu\cdot x\in K(M)$. Очевидно, для любого $\mu=1+\xi^2\mu',\ \ \mu'\in E_2'(\xi)$ соответствующий автомат также есть в $K(M):\ x+\xi^2\mu'\cdot x=(1+\xi^2\mu')x\in K(M)$.

Заметим, что любую константу из $E_2'(\xi)$ можно представить так: $\mu=a_0+a_1\xi+\xi^2\mu',\ \mu'\in E_2'(\xi),\ a_0,a_1\in E_2$. Подставив автомат f_3 в автоматы, представленные выше, получим любую константу вида $(a_0+0\cdot\xi+\xi^2\mu')$. Если в сумматор от двух переменных подставить на один вход автомат f_4, а на другой вход — константы $(a_0+0\cdot\xi+\xi^2\mu')$, то получим в K-замыкании множества M любую константу из $E_2'(\xi)$.

Таким образом, в $K(M)$ есть любой автомат из M_1, т.е. $M_1\subset K(M)$. И в силу написанного выше получаем, что $K(M)=M_1$.

Покажем, что множество M — это базис класса автоматов M_1. Автомат f_1 является единственным автоматом более, чем от одной переменной, поэтому $K(\{f_2,f_3,f_4\})\neq M_1$. f_1,f_2,f_4 принадлежат K-замкнутому классу T_0, т.е. $K(\{f_1,f_2,f_4\})\neq M_1$, а автомат $f_3\notin T_0$. Автомат с нулевым свободным членом f_2 можно получить только применяя операции композиции к автомату с нулевым свободным членом f_1. Коэффициенты автоматов f_1,f_2 в силу свойств класса M_1 принадлежат $K^{(1)}$-замкнутому классу $M_1^{(1)}$. Несложно показать, что $\xi^2\notin K^{(1)}(\{1,\xi^3\})$, то получить автомат f_2 в K-замыкании множества $\{f_1,f_3,f_4\}$ нельзя.

Заметим, что, поскольку автоматы f_1,f_2 имеют нулевой свободный член, то и в их K-замыкании можно получить только автомат с нулевым свободным членом. Т.е. используя операции композиции и автоматы из $K(\{f_1,f_2\})$, получить константу можно только, если подставить некоторую константу в автомат из $K(\{f_1,f_2\})$. С другой стороны, так как все коэффициенты при переменных этих автоматов $U(\{f_1,f_2\})=$ $=\{1,\xi^2,\xi^3\}$ принадлежат $K^{(1)}$-замкнутому классу $M_1^{(1)}$, то и в $K(\{f_1,f_2\})$

автоматы будут иметь коэффициенты при переменных из $M_1^{(1)}$ (в силу замкнутости соответствующих классов). Следовательно, в $K(\{f_1, f_2, f_3\})$ есть только константы из $M_1^{(1)}$. А, т.к. $\xi \notin M_1^{(1)}$, то и константы ξ в K-замыкании множества $\{f_1, f_2, f_3\}$ нет.

Ввиду выше изложенного, приходим к выводу, что множество M является базисом K-замкнутого класса M_1. Лемма полностью доказана.

□

Теорема 1. *Каждый предполный класс линейных автоматов, входящий в A-критериальную систему, является K-конечнопорожденным.*

Доказательство. $\mathcal{J}_A = \{T_0, T_1, V_1, V_2, M_1\}$ — A-критериальная система в L_2. В леммах 1—5 было доказано, что данные классы являются K-конечнопорожденными. Теорема доказана.

□

Следствие 1. *Каждый предполный класс линейных автоматов, входящий в A-критериальную систему, является A-конечнопорожденным.*

Доказательство. Поскольку оператор A-замыкания сильнее оператора K-замыкания, то из K-конечнопорожденности следует A-конечнопорожденность. В теореме 1 было доказано, что каждый класс A-критериальной системы является K-конечнопорожденным и, следовательно, он является A-конечнопорожденным классом автоматов. Следствие доказано.

□

4. Заключение

Таким образом, в настоящей работе был рассмотрен вопрос K- и A- конечнопорожденноти для предполных классов в классе линейных автоматов, функционирующих на полем Галуа, состоящим из двух элементов. Для всех предполных классов, образующих А-критериальную систему было доказано, что они являются К-конечнопорожденными и, следовательно, А-конечнопорожденными. Также для каждого из этих классов был предъявлен базис. В дальнейших публикациях будут представлены результаты решения подобных задач для других предполных классов линейных автоматов.

Автор выражает благодарность своему научному руководителю доценту кафедры МаТИС механико-математического факультета МГУ, д.ф.-м.н. Часовских Анатолию Александровичу за постановку задачи и помощь в исследовании.

Список литературы

[1] Кудрявцев В. Б., Алешин С. В., Подколзин А. С., *Введение в теорию автоматов*, Наука, Москва, 1985, 320 с.

[2] Клини С. К., "Представление событий в нервных сетях и конечных автоматах", *Автоматы*, 1956, С. 15-67.

[3] Мур Э. Ф., "Умозрительные эксперименты с последовательностными машинами", *Автоматы*, 1956, С. 179-210.

[4] Нейман Дж., "Вероятностная логика и синтез надежных организмов из ненадежных компонент", *Автоматы*, 1956, С. 68-139.

[5] Яблонский С. В., "О построении тупиковых кратных экспериментов для автоматов", *Тр. МИАН СССР*, **133** (1973), С. 263—272.

[6] Лупанов О. Б., "О сравнении двух типов конечных источников", *Проблемы кибернетики*, **9** (1963), С. 321–326.

[7] Кудрявцев В. Б., "Теорема полноты для одного класса автоматов без обратных связей", *Проблемы кибернетики*, **8** (1962), С. 91-115.

[8] Кудрявцев В. Б., "О мощности множеств предполных множеств некоторых функциональных систем, связанных с автоматами", *Проблемы кибернетики*, **13** (1965), С. 45–74.

[9] Часовских А. А., "О полноте в классе линейных автоматов", *Математические вопросы кибернетики*, **3** (1991), С. 140—166.

[10] Часовских А. А., "Проблема А-полноты линейно-автоматных функций над конечным полем", *Интеллектуальные системы. Теория и приложения*, **18**:1 (2014), С. 253—257.

[11] Часовских А. А., "Проблема полноты в классах линейных автоматов", *Интеллектуальные системы. Теория и приложения*, **22**:2 (2018), С. 151-154.

[12] Часовских А. А., "Максимальные подклассы в классах линейных автоматов над конечными полями", *Дискретная математика*, **31**:4 (2019), С. 88-101.

[13] Гилл, А., *Линейные последовательностные машины*, Наука, Москва, 1974, 288 с.

[14] Лидл Р., Нидеррайтер Г., *Конечные поля: в 2 т. Т. 1.*, Мир, Москва, 1988, 430 с.

The problem of K-finite generation for precomplete classes of linear automata constituting an A-criterion system in the space of linear automata.

Biryukova Veronika Andreevna

This article considers the problem of K- and A-finite generation for precomplete classes of linear automata operating over the Galois field consisting of two elements. The set of all studied classes is the A-criterion system in the class of linear automata. A finite basis was presented for each class under consideration.

Keywords: finite automaton, linear automaton, composition operation, feedback operation, completeness, closed class, precomplete class, K-finitely generated class, A-finitely generated class.

References

[1] Kudryavtsev V.B., Alyoshin S.V., Podkolzin A.S., *Introduction to automata theory*, «Science», Moscow, 1985 (In Russian), 320 c.

[2] S.C. Kleene, "Representation of events in nerve nets and finite automata", *Automata Studies*, 1956, 15-67 (In Russian).

[3] E. F. MOORE, "Gedanken-experiments on sequential machines", *Automata Studies*, 1956, 179-210 (In Russian).

[4] J. von Neumann, "Probabilistic logics and the synthesis of reliable organisms from unreliable components", *Automata Studies*, 1956, 68-139 (In Russian).

[5] S. Yablonsky, "On the construction of dead-end multiple experiments for automata", *Proceedings of the Steklov Institute of Mathematics*, **133** (1973), 263—272 (In Russian).

[6] O. Lupanov, "About comparing two types of finite sources", *Problems of cybernetics*, **9** (1963), 321–326 (In Russian).

[7] V. Kudryavtsev, "Completeness theorem for one class of automata without feedback", *Problems of cybernetics*, **8** (1962), 91-115 (In Russian).

[8] V. Kudryavtsev, "The Cardinality of Sets of Precomplete Sets of Some Functional Systems Related to Automata", *Problems of cybernetics*, **13** (1965), 45–74 (In Russian).

[9] Chasovskih A.A, "Completeness in the class of linear automata", *Mathematical problems of cybernetics*, **3** (1991), 140—166 (In Russian).

[10] Chasovskih A.A, "The problem of A-completeness of linear automaton functions over a finite field", *Intelligent Systems. Theory and Applications*, **18**:1 (2014), 253—257 (In Russian).

[11] Chasovskih A.A, "The problem of completeness in classes of linear automata", *Intelligent Systems. Theory and Applications*, **22**:2 (2018), 151-154 (In Russian).

[12] Chasovskih A.A., "Maximal subclasses in classes of linear automata over finite fields", *Discrete Math*, **31**:4 (2019), 88-101 (In Russian).

[13] A. Gill, *Linear Sequential Circuits*, Science, Moscow, 1974 (In Russian), 288 c.

[14] R. Lidl, H. Niederreiter, *Finite Fields, volume 1*, 'World', Moscow, 1988 (In Russian), 430 c.

Сложение векторов на прямой с помощью клеточного автомата с локаторами

Д. Э. Ибрагимова[1]

В данной статье рассматривается применение модели клеточного автомата с локаторами к задаче сложения векторов на прямой. Модель клеточного автомата с локаторами подразумевает возможность каждой ячейки автомата передавать сигнал на сколь угодно большие расстояния. В статье показано, что эта возможность позволяет уменьшить сложность рассматриваемой задачи с линейной до логарифмической по сравнению с классической моделью клеточного автомата.

Ключевые слова: клеточные автоматы, однородные структуры, сложение векторов.

1. Введение

Клеточные автоматы являются дискретными математическими моделями широкого класса реальных систем вместе с протекающими в них процессами.

Понятие клеточного автомата возникло в результате усовершенствования модели Дж. фон Неймана [1, 2, 3], предложенной им для описания процессов самовоспроизведения в биологии и технике. Эта модель развивалась и использовалась в работах А. Беркса [4], Э. Мура [5], В. Б. Кудрявцева, А. С. Подколзина, А. А. Болотова [6] и других исследователей.

Клеточный автомат — это математический объект с дискретными пространством и временем. Пространство поделено на клетки, в каждой из которых находится элементарный автомат. Такты времени задаются натуральными числами. Состояние каждой пространственной клетки определяется очень простыми правилами взаимодействия. Эти правила предписывают изменения состояния каждой клетки в следующем такте времени в ответ на текущее состояние автоматов соседних клеток.

В работе Гасанова Э.Э. [7] было введено понятие клеточного автомата с локаторами, которое отличается от понятия обычного клеточного автомата тем, что допускает передачу информации не только между

[1] *Ибрагимова Дильноза Эльдаровна* — старший преподаватель каф. прикладной математики и информатики филиала МГУ имени М. В. Ломоносова в г. Ташкенте, e-mail: ibragimova70@mail.ru.

Ibragimova Dilnoza Eldarovna — senior lecturer, Tashkent Branch of Lomonosov Moscow State University, Chair of Applied Mathematics and Computer Science.

соседними ячейками, но и на любое расстояние, посредством передачи сигнала в эфир. В работе рассматривается применение этой модели к одномерной задаче сложения векторов: на [1] произвольно отмечены особая точка, называемая "началом координат" и две других, которые будем называть "концами векторов". Нужно найти точку расстояние до которой от "начала координат" будет равно сумме расстояний от "начала координат" до "концов векторов".

Классическая модель клеточного автомата решает эту задачу за не менее чем линейное (по минимальному расстоянию между началом координат и концами векторов) время. В данной работе будет показано, что модель автомата с локаторами может решать эту задачу за логарифмическое время.

Автор выражает благодарность профессору Э.Э.Гасанову за постановку задачи.

2. Постановка задачи и формулировка результата

Приведем понятие клеточного автомата с локаторами, которое ранее было введено в работе Э. Э. Гасанова [7]. Приводимое здесь определение клеточного автомата с локаторами учитывает замечания Г. В. Калачева [8] и ранее в таком виде в литературе не встречалось.

Под *телесным углом* в $\mathbb{R}^k$ будем понимать часть пространства $\mathbb{R}^k$, которая является объединением всех лучей, выходящих из данной точки (*вершины угла*) и пересекающих некоторую гиперповерхность в $\mathbb{R}^k$. По определению будем считать, что вершина телесного угла не входит в телесный угол.

Рациональным телесным углом будем называть телесный угол, границы которого являются частями гиперплоскостей, задаваемых линейными уравнениями с целыми коэффициентами.

Клеточным автоматом с локаторами называется восьмерка $\sigma = (\mathbb{Z}^k, Q, V, G, +, L, \varphi, \psi)$, где $\mathbb{Z}^k$ — множество k-мерных векторов с целыми координатами, Q — некоторое конечное множество, называемое *множеством состояний*; в множестве Q выделено одно состояние q_0, называемое *состоянием покоя*; $V = (\alpha_1, \ldots, \alpha_{h-1})$ — упорядоченный набор попарно различных векторов из $\mathbb{Z}^k$; G — коммутативная полугруппа с нейтральным элементом e; $+$ — коммутативная полугрупповая операция заданная на G; $L = (\nu_1, \ldots, \nu_m)$ — упорядоченный набор попарно различных рациональных телесных углов в $\mathbb{R}^k$ с вершиной в начале координат; φ — функция, зависящая от переменных $x_0, x_1, \ldots, x_{h-1}, z_1, \ldots, z_m$; $\varphi : Q^h \times G^m \to Q$, $\varphi(\mathbf{q_0}, \mathbf{e}) = q_0$; $\mathbf{q_0} = (q_0, \ldots, q_0) \in Q^h$, $\mathbf{e} = (e, \ldots, e) \in G^m$; ψ — функция зависящая от переменных $x_0, x_1, \ldots, x_{h-1}, z_1, \ldots, z_m$; $\psi : Q^h \times G^m \to G$; $\psi(\mathbf{q_0}, \mu) = e$, $\mu \in G^m$. Элементы множества $\mathbb{Z}^k$ называ-

ются *ячейками* клеточного автомата σ; элементы множества Q называются *состояниями ячейки* клеточного автомата σ; набор V называется *шаблоном соседства* клеточного автомата σ; элементы множества G называются *сигналами вещания*; набор L называется *шаблоном локаторов* клеточного автомата σ; функция φ называется *локальной функцией переходов* автомата σ; функция ψ называется *функцией вещания* автомата σ; переменные $x_0, x_1, \ldots, x_{h-1}$ принимают значения из Q, переменные $z_1, \ldots, z_m$ принимают значения из G. Состояние q_0 интерпретируется как *состояние покоя*, а условие $\varphi(\mathbf{q_0}, \mathbf{e}) = q_0$ — как *условие сохранения состояния покоя*. Ячейки, находящиеся в состоянии отличном от q_0, будем называть *активными*. Условие $\psi(\mathbf{q_0}, \mu) = e$ означает, что ячейка в состоянии покоя и не имеющая активных соседей посылает в эфир нейтральный элемент, что можно интерпретировать как то, что она не посылает сигналы в эфир.

Здесь нам нужно было вводить упорядочение шаблона соседства V и шаблона локаторов L для того, чтобы установить взаимно однозначное соответствие между векторами из V и телесными углами из L и переменными локальной функции переходов φ и функции вещания ψ соответственно $x_0, x_1, \ldots, x_{h-1}$ и $z_1, \ldots, z_m$. Это соответствие можно сделать более явным, если индексировать переменные функций φ и ψ самими векторами и телесными углами, т.е. считать, что локальная функция переходов φ и функции вещания ψ зависят от переменных $x_{\mathbf{0}}, x_{\alpha_1}, \ldots, x_{\alpha_{h-1}}, z_{\nu_1}, \ldots, z_{\nu_m}$, здесь индекс первой переменной есть нулевой вектор $\mathbf{0} = (0, \ldots, 0) \in \mathbb{Z}^k$. Если договориться так индексировать переменные локальной функции переходов и функции вещания, то их можно записывать в любом порядке, и тогда можно воспринимать шаблон соседства и шаблон локаторов как просто множества, а не упорядоченный набор. В дальнейшем мы будем индексировать переменные локальной функции переходов и функции вещания векторами из шаблона соседства и телесными углами из шаблона локаторов.

Если $\alpha \in \mathbb{Z}^k$ и ν — телесный угол с вершиной в начале координат, то через $\nu(\alpha)$ обозначим телесный угол, полученный параллельным переносом телесного угла ν на вектор α, т.е. вершиной телесного угла $\nu(\alpha)$ является точка α.

Если $\alpha \in \mathbb{Z}^k$ — ячейка клеточного автомата σ, то множество $V(\alpha) = \{\alpha, \alpha+\alpha_1, \ldots, \alpha+\alpha_{h-1}\}$ называется *окрестностью ячейки* α, а множество $L(\alpha) = \{\nu_1(\alpha), \ldots, \nu_m(\alpha_m)\}$ называется *локаторами ячейки* α.

Состоянием клеточного автомата с локаторами σ назовем пару (g, f), где g — произвольная функция, определенная на множестве Z^k, принимающая значения из G, называемая *состоянием эфира*, f — произвольная функция, определенная на множестве Z^k, принимающая значения из Q и называемая *распределением состояний клеточного автома-*

та с локаторами σ. Такую пару функций можно интерпретировать как некую мозаику, получающуюся в k-мерном пространстве приписыванием каждой точке с целочисленными координатами некоторого сигнала из G и некоторого состояния из Q. Множество всевозможных состояний клеточного автомата с локаторами обозначим Σ.

Если $\alpha \in \mathbb{Z}^k$, (g, f) — состояние клеточного автомата с локаторами σ, то значение $g(\alpha)$ назовем *сигналом ячейки* α, определяемым состоянием (g, f), а значение $f(\alpha)$ — *состоянием ячейки* α, определяемым состоянием (g, f).

Для каждого $i \in \{1, \ldots, m\}$

$$s_i(\alpha) = \sum_{\beta \in \nu_i(\alpha) \cap \mathbb{Z}^k} g(\beta) \tag{1}$$

назовем *значением локатора* ν_i, определяемым состоянием (g, f). Здесь суммирование сигналов осуществляется с помощью определяющей операции + полугруппы G.

На множестве Σ определим *глобальную функцию переходов* Φ клеточного автомата с локаторами σ, полагая $\Phi(g, f) = (g', f')$, где $(g, f), (g', f') \in \Sigma$ и для любой ячейки $\alpha \in \mathbb{Z}^k$ выполняются тождества

$$f'(\alpha) = \varphi(f(\alpha), f(\alpha + \alpha_1), \ldots, f(\alpha + \alpha_{h-1}), s_1(\alpha), \ldots, s_m(\alpha)), \tag{2}$$

$$g'(\alpha) = \psi(f(\alpha), f(\alpha + \alpha_1), \ldots, f(\alpha + \alpha_{h-1}), s_1(\alpha), \ldots, s_m(\alpha)). \tag{3}$$

Содержательная интерпретация отображения Φ такова, что сигнал каждой ячейки и состояние каждой ячейки "после перехода" определяется по состоянию упорядоченной окрестности ячейки и по значениям локаторов "до перехода" с помощью законов φ и ψ одинаково для всех ячеек.

Поведениями клеточного автомата с локаторами σ назовем такие последовательности $(g_0, f_0), (g_1, f_1), (g_2, f_2), \ldots$ его состояний, для которых выполняется $(g_{i+1}, f_{i+1}) = \Phi(g_i, f_i)$ для всех $i = 0, 1, 2, \ldots$, причем (g_i, f_i) называется *состоянием клеточного автомата с локаторами* σ *в момент* i, а (g_0, f_0) называется *начальным состоянием клеточного автомата с локаторами* σ.

Сформулируем задачу сложения векторов на прямой.

Пусть в пространстве $\mathbb{Z}^1$ задано начальное состояние I клеточного автомата, удовлетворяющее следующим условиям:

1) Любой ячейке присвоено одно из трех состояний $\{B, E, *\}$, где $*$ соответствует состоянию покоя, B интерпретируется как начало координат, а E — как концы векторов.

2) Есть лишь одна ячейка, которой присвоено состояние B, и не больше двух ячеек, которым присвоено состояние E.

Решением задачи сложения векторов на прямой, соответствующей начальному состоянию I, назовем состояние автомата, называемое финальным и удовлетворяющее следующим условиям:

1) Ячейке, которой в начальном состоянии было присвоено состояние B, присвоено одно из двух состояний B_F или $*$, в зависимости от расположения ячеек в состоянии E. Если в I ячейки E располагаются на равном расстоянии слева и справа от ячейки B, то в этом случае сумма векторов будет равна 0 и ячейка B перейдет в состояние $*$. Иначе ячейка B перейдет в состояние B_F.

2) Ячейке, которая находится от ячейки B на расстоянии равном сумме векторов, присвоено состояние E_F.

3) Ячейкам, которые находятся между ячейками в состояниях B_F и E_F, присвоено состояние C.

4) Остальным ячейкам присвоено состояние $*$.

5) Описанное выше финальное состояние в дальнейшем не изменяется.

Справедлива следующая теорема.

Теорема 1. *Существует клеточный автомат с локаторами σ с 38 состояниями и с мощностью алфавита вещания 40, который решает задачу сложения векторов на прямой за время не превышающее $2\log_2 n + 6$, где n — расстояние от ячейки с начальным состоянием B до ближайшей ячейки с начальным состоянием E.*

3. Доказательство теоремы

Рассмотрим клеточный автомат $\sigma = (\mathbb{Z}^1, Q, V = \{\varnothing\}, G, +, L, \varphi, \psi)$, где $G = \{0,1\}^3 \times \{0,1,2,3,4\}$, а $L = \{\nu_{-1}, \nu_1\}$, где ν_{-1}, ν_1 — телесные углы, соответствующие векторам (-1) и (1), т.е. шаблон локаторов состоит из двух лучей направленных влево и вправо.

Полугрупповую операцию на G определим следующим образом:

$$(a_1, b_1, c_1, d_1) + (a_2, b_2, c_2, d_2) = (a_1 \oplus a_2, b_1 \oplus b_2, max(c_1, c_2), max(d_1, d_2)).$$

По первой и второй компоненте сигнала из G суммирование берется по модулю 2. По первой компоненте передается побитово двоичная запись

длины более короткого вектора. По второй компоненте эта длина откладывается от конца более длинного вектора. По третьей и четвертой компоненте ячейки передают сигналы о своем состоянии.

Нейтральным элементом данной полугруппы G является элемент $(0,0,0,0)$.

Определим также множество состояний

$$Q = \{*, B, B_B, B_1, B_*, B_C, B_>, B_<, B_=, B_R, B_2, E, E_1, E_{1-}, E_E,$$
$$L_+, L_-, L_C, L_1, L_{1-}, L_0, L_L, L_*, L_2, R_+, R_-, R_C, R_1, R_{1-},$$
$$R_{1C}, R_{0C}, R_R, R_*, R_2, B_F, E_F, C, S\}.$$

Состояние $*$ соответствует состоянию покоя.

Сначала приведем неформальное описание алгоритма решения задачи. В случае когда оба вектора направлены в одну сторону, все активные ячейки находящиеся между началом и концом меньшего вектора посылают сигнал (1,0,1,0). Полугрупповая операция определена таким образом, что по первой компоненте вычисляется сумма по модулю два. Ячейки, которые находятся между началом и концом меньшего вектора, услышав по первой компоненте сигнала вещания 0, становятся молчащими. Таким образом, каждая вторая активная ячейка переходит в молчащее состояние. Ячейка, являющаяся концом меньшего вектора, посылает по четвертой компоненте сигнал, который услышала по первой компоненте слева (справа). То есть, конец вектора передает двоичную запись длины меньшего вектора. Все активные ячейки, находящиеся правее(левее) конца большего вектора, посылают сигнал (0,1,2,0). Этих ячеек бесконечное количество, так как мы не можем заранее знать длину результирующего вектора. По второй компоненте также вычисляется сумма по модулю два. Для ячеек, находящихся правее (левее) конца большего вектора, переход в молчащее состояние осуществляется по следующему принципу, если ячейка слышит разные сигналы по второй и четвертой компонентам сигнала вещания, то переходит в молчащее состояние. Таким образом, каждая вторая активная ячейка правее (левее) конца большего вектора переходит в молчащее состояние. По третьей и четвертой компоненте ячейки слышат самый сильный сигнал. Если ячейка, являющаяся концом меньшего вектора, слышит слева (справа) сигнал (0,0,0,0), то она посылает соответствующий сигнал о том, что не осталось ни одной активной ячейки между началом и концом меньшего вектора. Длина непрерывного отрезка из молчащих ячеек, начинающегося сразу после конца большего вектора, равна длине меньшего вектора. Значит самая левая(правая) активная ячейка, находящаяся правее(левее) конца большего вектора является концом результирующего вектора. Остальные активные ячейки, за исключением начала векторов, необходимо перевести в неактивное состояние или состояние покоя.

Для случая когда вектора направлены в противоположные стороны необходимо вначале определить направление результирующего вектора. Для этого все активные ячейки, находящиеся между концами и началом векторов посылают сигнал (1,0,1,0). Центральная ячейка сравнивает сигналы слева и справа поступающие по первой компоненте. И меняет свое состояние в зависимости от того с какой стороны слышит более сильный сигнал. Каждая вторая активная ячейка, находящаяся между началом и концом вектора, на каждом такте переходит в молчащее состояние. Когда с какой-либо из сторон не останется активных ячеек, это определяется по сигналу посылаемому ячейками по третьей компоненте, центральная ячейка, в зависимости от своего состояния, посылает соответствующий сигнал. По этому сигналу ячейки, находящиеся между концами и началом векторов, определяют в какое состояние перейти. Состояние активных ячеек определяет по сигналу с какой стороны ячейки будут переходить в молчащее состояние. Например, если положительный вектор больше отрицательного, то надо будет откладывать длину отрицательного вектора от конца положительного вектора в левую сторону. Значит ячейки будут переходить в молчащее состояние по сигналу, который слышат справа. Длина непрерывного отрезка из молчащих ячеек, которые расположены сразу за концом положительного вектора слева, в этом случае будет равна длине отрицательного вектора. Значит самая правая активная ячейка, из тех активных ячеек, что находятся между началом и концом положительного вектора, является концом результирующего вектора. В случае когда длина отрицательного вектора больше рассуждения аналогичны, и самая левая активная ячейка, из тех активных ячеек, что находятся между началом и концом отрицательного вектора, является концом результирующего вектора.

Опишем функции φ и ψ для каждого состояния автомата. Для простоты изложения будем называть ячейку, которая в начальный момент I имела состояние B, *началом векторов* или *центральной ячейкой*. А ячейки, которые в I имели состояние E, будем называть *концами векторов*. Если активная ячейка посылает в эфир сигнал $(0,0,0,0)$, будем говорить что она молчит. Пусть $z_{\nu_{-1}} = \{z^1_{\nu_{-1}}, z^2_{\nu_{-1}}, z^3_{\nu_{-1}}, z^4_{\nu_{-1}}\}$ и $z_{\nu_1} = \{z^1_{\nu_1}, z^2_{\nu_1}, z^3_{\nu_1}, z^4_{\nu_1}\}$.

Будем проводить доказательство теоремы, одновременно демонстрируя поведение автомата на примерах с начальным состоянием, приведенном ниже:

Пример 1. Случай, когда оба вектора направлены вправо.
$t = 0$

q	$*$	B	$*$	$*$	$*$	$*$	$*$	E	$*$	E	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$
ν_{-1}^1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ν_{-1}^2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ν_{-1}^3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ν_{-1}^4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ν_1^1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ν_1^2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ν_1^3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ν_1^4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ψ_1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ψ_2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ψ_3	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0
ψ_4	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Пример 2. Случай, когда вектора направлены в противоположные стороны.

$t = 0$

q	$*$	E	$*$	$*$	$*$	$*$	$*$	B	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	E	$*$
ν_{-1}^1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ν_{-1}^2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ν_{-1}^3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ν_{-1}^4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ν_1^1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ν_1^2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ν_1^3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ν_1^4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ψ_1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ψ_2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ψ_3	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
ψ_4	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0

Здесь q — строка состояний клеточного автомата; ν_{-1}^1, ν_{-1}^2, ν_{-1}^3, ν_{-1}^4 — строки, соответствующие компонентам значения локатора ν_{-1}; ν_1^1, ν_1^2, ν_1^3, ν_1^4 — строки, соответствующие компонентам значения локатора ν_1; ψ_1, ψ_2, ψ_3, ψ_4 — строки, соответствующие компонентам сигнала вещания, посылаемого в эфир.

B — это начальное состояние ячейки, определяющей начало векторов. Ячейка в этом состоянии посылает сигнал $(0, 0, 0, 1)$ и ждет сигнала от ячеек в состоянии E, для того чтобы определить находится ли она справа или слева от "концов векторов" или между ними. E — это начальное состояние ячейки, определяющее конец вектора. В этом состоянии ячейка посылает сигнал $(0, 0, 1, 0)$. $*$ — это начальное состояние остальных ячеек (не начала и не конца вектора). Ячейки в этом состоянии ждут сигнала от других ячеек, чтобы перейти в левую или правую версию.

Этап 1: Определение ориентации.

На первом такте сигналы подают только ячейки в состоянии B и E для того, чтобы остальные ячейки смогли определить свою ориентацию.

В примере 1 на втором такте ячейка B получив сигнал (0,0,1,0) только справа, понимает что оба конца вектора находятся справа и переходит в состояние B_B. В состояние B_B центральная ячейка переходит в случае когда оба конца векторов расположены слева или справа от начала

векторов. То есть либо оба вектора положительны, либо оба — отрицательны.

Ячейки в состоянии $*$, находящиеся левее начала векторов переходят в состояние R_-. Остальные ячейки, находящиеся в состоянии $*$ переходят в состояния L_+, R_+ или R_C.

Состояние L_+ — это состояние для всех ячеек, находящихся между концом вектора меньшей длины и началом векторов. Состояние R_+ — это состояние, которое принимают ячейки, находящиеся правее самого правого конца вектора. Состояние R_C — это состояние ячеек, находящихся между концами векторов.

$t = 1$

q	R_-	B_B	L_+	L_+	L_+	L_+	L_+	E	R_C	E	R_+	R_+	R_+	R_+	R_+	R_+	R_+	R_+
ν_{-1}^1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ν_{-1}^2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ν_{-1}^3	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1
ν_{-1}^4	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
ν_1^1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ν_1^2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ν_1^3	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0
ν_1^4	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ψ_1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ψ_2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ψ_3	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0
ψ_4	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

В примере 2 на втором такте ячейка в состоянии B, услышав слева и справа сигналы (0,0,1,0), понимает, что вектора направлены в противоположные стороны и переходит в состояние B_C. Ячейки, находящиеся между центральной ячейкой и концом положительного вектора, переходят в состояние R_C. Ячейки, находящиеся между центральной ячейкой и концом отрицательного вектора, переходят в состояние L_C. Ячейки, находящиеся правее конца положительного вектора, переходят в состояние R_+. Ячейки, находящиеся левее конца отрицательного вектора, переходят в состояние R_-.

$t = 1$

q	R_-	E	L_C	L_C	L_C	L_C	L_C	B_C	R_C	R_C	R_C	R_C	R_C	R_C	R_C	R_C	E	R_+
ν_{-1}^1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ν_{-1}^2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ν_{-1}^3	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
ν_{-1}^4	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1
ν_1^1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ν_1^2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ν_1^3	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0
ν_1^4	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0
ψ_1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ψ_2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ψ_3	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
ψ_4	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	0	0

Опишем функции φ и ψ для состояний B и $*$.

$$\varphi(B, z_{\nu_{-1}}, z_{\nu_1}) = \begin{cases} B_B, \text{если } (z^3_{\nu_{-1}} = 0,\ z^4_{\nu_{-1}} = 0,\ z^3_{\nu_1} = 1,\ z^4_{\nu_1} = 0) \\ \qquad \text{или } (z^3_{\nu_{-1}} = 1,\ z^4_{\nu_{-1}} = 0,\ z^3_{\nu_1} = 0,\ z^4_{\nu_1} = 0); \\ B_C,\ \text{если } z^3_{\nu_{-1}} = 1,\ z^4_{\nu_{-1}} = 0,\ z^3_{\nu_1} = 1,\ z^4_{\nu_1} = 0; \end{cases}$$

$$\psi(B, z_{\nu_{-1}}, z_{\nu_1}) = (0, 0, 0, 1).$$

$$\varphi(*, z_{\nu_{-1}}, z_{\nu_1}) = \begin{cases} R_-,\ \text{если } z^3_{\nu_{-1}} = z^4_{\nu_{-1}} = 0,\ z^3_{\nu_1} = z^4_{\nu_1} = 1; \\ R_+,\ \text{если } z^3_{\nu_{-1}} = z^4_{\nu_{-1}} = 1,\ z^3_{\nu_1} = z^4_{\nu_1} = 0; \\ L_C,\ \text{если } z^3_{\nu_{-1}} = 1,\ z^4_{\nu_{-1}} = 0,\ z^3_{\nu_1} = z^4_{\nu_1} = 1; \\ R_C,\ \text{если } z^3_{\nu_{-1}} = z^4_{\nu_{-1}} = 1,\ z^3_{\nu_1} = 1,\ z^4_{\nu_1} = 0; \\ L_-,\ \text{если } z^3_{\nu_{-1}} = 1,\ z^4_{\nu_{-1}} = 0,\ z^3_{\nu_1} = 0,\ z^4_{\nu_1} = 1; \\ L_+,\ \text{если } z^3_{\nu_{-1}} = 0,\ z^4_{\nu_{-1}} = 1,\ z^3_{\nu_1} = 1,\ z^4_{\nu_1} = 0. \end{cases}$$

$$\psi(*, z_{\nu_{-1}}, z_{\nu_1}) = (0, 0, 0, 0).$$

В случае когда оба вектора отрицательные, ячейки, находящиеся правее самой правой ячейки в состоянии E, переходят в состояние R_-. Ячейки, находящиеся между концом вектора меньшей длины и началом векторов, переходят в состояние L_-. Ячейки, находящиеся между концами векторов, переходят в состояние L_C.

Состояния L_+, L_-, R_+, R_-, L_C и R_C — это, по сути, состояния сна. Эти состояния нужны для того, чтобы центральная ячейка успела определить свое положение относительно концов векторов и послала соответствующий сигнал. По этому сигналу остальные ячейки могут определиться со своей правой или левой версией.

В примере 1 ячейка B_B посылает сигнал (0,0,0,2) для того, чтобы концы векторов определили свое положение относительно начала векторов.

$$\varphi(B_B, z_{\nu_{-1}}, z_{\nu_1}) = B_1.$$

$$\psi(B_B, z_{\nu_{-1}}, z_{\nu_1}) = \begin{cases} (0, 0, 0, 2),\ \text{если } z^3_{\nu_{-1}} = 0,\ z^4_{\nu_{-1}} = 0,\ z^3_{\nu_1} = 1,\ z^4_{\nu_1} = 0; \\ (0, 0, 0, 4),\ \text{если } z^3_{\nu_{-1}} = 1,\ z^4_{\nu_{-1}} = 0,\ z^3_{\nu_1} = 0,\ z^4_{\nu_1} = 0. \end{cases}$$

Сигнал (0,0,0,4) ячейка B_B посылает в случае, когда оба вектора отрицательные.

В примере 2 ячейка B_C посылает сигнал (0,0,0,3).

$$\varphi(B_C, z_{\nu_{-1}}, z_{\nu_1}) = B_=.$$

$$\psi(B_C, z_{\nu_{-1}}, z_{\nu_1}) = (0, 0, 0, 3).$$

Ячейка E, во всех случаях, ждет сигнала от начала векторов, чтобы определить свое положение относительно начала векторов, находится ли она слева или справа от начала векторов. То есть "вектора" определяют свою ориентацию, отрицательную или положительную.

$$\varphi(E, z_{\nu_{-1}}, z_{\nu_1}) = \begin{cases} E_{1-}, \text{ если } (z^3_{\nu_{-1}} = 1,\ z^4_{\nu_{-1}} = 0,\ z^3_{\nu_1} = 0,\ z^4_{\nu_1} = 4) \\ \qquad \text{или } (z^3_{\nu_{-1}} = z^4_{\nu_{-1}} = 0,\ z^3_{\nu_1} = 0,\ z^4_{\nu_1} = 4); \\ E_1, \text{ если } (z^3_{\nu_{-1}} = 0,\ z^4_{\nu_{-1}} = 2,\ z^3_{\nu_1} = 1,\ z^4_{\nu_1} = 0) \\ \qquad \text{или } (z^3_{\nu_{-1}} = 0,\ z^4_{\nu_{-1}} = 2,\ z^3_{\nu_1} = z^4_{\nu_1} = 0); \\ E_E, \text{ если } (z^3_{\nu_{-1}} = z^4_{\nu_{-1}} = 0,\ z^3_{\nu_1} = 1,\ z^4_{\nu_1} = 3) \\ \qquad \text{или } (z^3_{\nu_{-1}} = 1,\ z^4_{\nu_{-1}} = 3,\ z^3_1 = 0,\ z^4_1 = 0); \\ C, \text{ если } (z^3_{\nu_{-1}} = 1,\ z^4_{\nu_{-1}} = 2,\ z^3_{\nu_1} = z^4_{\nu_1} = 0) \\ \quad \text{или } (z^3_{\nu_{-1}} = z^4_{\nu_{-1}} = 0,\ z^3_{\nu_1} = 0,\ z^4_{\nu_1} = 4). \end{cases}$$

$$\psi(E, z_{\nu_{-1}}, z_{\nu_1}) = (0, 0, 1, 0).$$

Состояние E_{1-} возникает когда оба вектора отрицательные. Состояние E_1 возникает когда оба вектора положительные. Состояние E_E возникает когда вектора направлены в противоположные стороны.

На этом этап определения ориентации заканчивается.

Этап 2. Сравнение длин векторов, если вектора направлены в разные стороны.

В случае когда концы векторов расположены по разные стороны от центральной ячейки (пример 2), прежде чем находить сумму векторов, надо определить направление результирующего вектора. Для этого надо определить какой из векторов длиннее. Алгоритм сравнения длин отрезков описан в статье Васильева Д. И.[9]. В случае, когда вектора расположены по одну сторону от центральной точки (пример 1) этап сравнения длин пропускается.

Для примера 2 на третьем такте ячейки L_C переходят в состояние L_1.

$$\varphi(L_C, z_{\nu_{-1}}, z_{\nu_1}) = \begin{cases} C, \text{ если } z^3_{\nu_{-1}} = 1,\ z^4_{\nu_{-1}} = 0,\ z^3_{\nu_1} = 1,\ z^4_{\nu_1} = 4; \\ L_1, \text{ в остальных случаях.} \end{cases}$$

$$\psi(L_C, z_{\nu_{-1}}, z_{\nu_1}) = (0, 0, 0, 0).$$

Ячейки R_C переходят в состояние R_{1C}.

$$\varphi(R_C, z_{\nu_{-1}}, z_{\nu_1}) = \begin{cases} C, \text{ если } z^3_{\nu_{-1}} = 1,\ z^4_{\nu_{-1}} = 2,\ z^3_{\nu_1} = 1,\ z^4_{\nu_1} = 0; \\ R_{1C}, \text{ в остальных случаях.} \end{cases}$$

$$\psi(R_C, z_{\nu_{-1}}, z_{\nu_1}) = (0,0,0,0).$$

Ячейки R_+ и R_- переходят в состояние S. Состояние S — это состояние псевдопокоя. Все ячейки левее конца отрицательноговектора и все ячейки правее конца положительного вектора должны стать молчащими. В дальнейшем процессе они участвовать не будут.

$$\varphi(R_+, z_{\nu_{-1}}, z_{\nu_1}) = \begin{cases} R_1, \text{ если } z^4_{\nu_{-1}} = 0; \\ R_+, \text{ если } z^4_{\nu_{-1}} = 2; \\ S, \text{ если } z^4_{\nu_{-1}} = 3 \text{ или } z^4_{\nu_{-1}} = 4. \end{cases}$$

$$\psi(R_+, z_{\nu_{-1}}, z_{\nu_1}) = (0,0,0,0).$$

$$\varphi(R_-, z_{\nu_{-1}}, z_{\nu_1}) = \begin{cases} R_{1-}, \text{ если } z^4_{\nu_1} = 0; \\ R_-, \text{ если } z^4_{\nu_1} = 4; \\ S, \text{ если } z^4_{\nu_1} = 2 \text{ или } z^4_{\nu_1} = 3. \end{cases}$$

$$\psi(R_-, z_{\nu_{-1}}, z_{\nu_1}) = (0,0,0,0).$$

Концы векторов переходят в состояние E_E. Ячейка B_C переходит в состояние $B_=$.

$t = 2$

q	S	E_E	L_1	L_1	L_1	L_1	L_1	$B_=$	R_1C	R_1C	R_1C	R_1C	R_1C	R_1C	R_1C	R_1C	R_1C	E_E	S
ν^1_{-1}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ν^2_{-1}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ν^3_{-1}	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
ν^4_{-1}	0	0	0	0	0	0	0	0	3	3	3	3	3	3	3	3	3	3	3
ν^1_1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ν^2_1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ν^3_1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0
ν^4_1	3	3	3	3	3	3	3	0	0	0	0	0	0	0	0	0	0	0	0
ψ_1	0	0	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	0	0
ψ_2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ψ_3	0	0	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	0	0
ψ_4	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

Состояние L_1 — это состояния в какой-то момент всех ячеек, находящихся между началом и концом вектора меньшей длины, в случае когда оба вектора положительны. Или между центральной ячейкой и концом отрицательного вектора, в случае когда вектора направлены в противоположные стороны. На каждом следующем такте каждая вторая ячейка находящаяся в этом состоянии переходит в состояние S или состояние L_0. В данном случае переходят в состояние L_0.

$$\varphi(L_1, z_{\nu_{-1}}, z_{\nu_1}) = \begin{cases} L_0, \text{ если } z^1_{\nu_{-1}} = 0, \ z^4_{\nu_{-1}} = z^4_{\nu_1} = 1; \\ L_*, \text{ если } z^4_{\nu_1} = 3; \\ S, \text{ если } z^1_{\nu_{-1}} = 0, \ z^4_{\nu_{-1}} = 0, \ z^4_{\nu_1} = 0. \end{cases}$$

$$\psi(L_1, z_{\nu_{-1}}, z_{\nu_1}) = (1, 0, 1, 0).$$

В примере 2 ячейки в состоянии E_E посылают сигнал по четвертой компоненте для того, чтобы, после окончания этапа сравнения длин, определить которые из молчащих ячеек перевести в "кричащее" состояние.

$$\varphi(E_E, z_{\nu_{-1}}, z_{\nu_1}) = \begin{cases} E_E, \text{ если } z^4_{\nu_{-1}} = 0; \\ \text{S, если } z^4_{\nu_{-1}} = 2 \text{ или } z^4_{\nu_1} = 2. \end{cases}$$

$$\psi(E_E, z_{\nu_{-1}}, z_{\nu_1}) = (0, 0, 0, 1).$$

Центральная ячейка побитово сравнивает длины векторов. Побитовые записи длин векторов подаются на центральную ячейку от младшего разряда к старшему. Сравнивая соответствующие биты центральная ячейка переходит в одно из трех состояний:

- $B_<$ — если левый бит текущего разряда меньше правого. Если сигналы слева и справа равны, то состояние центральной ячейки не меняется;
- $B_>$ — если левый бит больше правого. Если сигналы слева и справа равны, то состояние центральной ячейки не меняется;
- $B_=$ — может возникнуть только если самые младшие разряды равны или если совпадают все биты, полученные центральной ячейкой.

$$\varphi(B_=, z_{\nu_{-1}}, z_{\nu_1}) = \begin{cases} B_>, \text{ если } z^1_{\nu_{-1}} = 1, \ z^1_{\nu_1} = 0; \\ B_<, \text{ если } z^1_{\nu_{-1}} = 0, \ z^1_{\nu_1} = 1; \\ S, \text{ если слышит сигнал (0,0,0,0) слева и справа.} \end{cases}$$

$$\psi(B_=, z_{\nu_{-1}}, z_{\nu_1}) = \begin{cases} (0, 0, 0, 4), \text{ если } z^3_{\nu_{-1}} = z^3_{\nu_1} = 0; \\ (0, 0, 0, 0), \text{ в остальных случаях.} \end{cases}$$

Дальше на каждом такте каждая вторая ячейка в состоянии R_{1C} будет переходить в состояние R_{0C}, а каждая вторая ячейка в состоянии L_1 будет переходить в состояние L_0.

$$\varphi(R_{1C}, z_{\nu_{-1}}, z_{\nu_1}) = \begin{cases} R_{0C}, \text{ если } z^1_{\nu_1} = 0, \ z^4_{\nu_{-1}} = z^4_{\nu_1} = 1; \\ R_*, \text{ если } z^4_{\nu_{-1}} = 2. \end{cases}$$

$$\psi(R_{1C}, z_{\nu_{-1}}, z_{\nu_1}) = (1, 0, 1, 0).$$

Состояние R_* мы опишем позже.

Сигнал по третьей компоненте ячейки L_1 и R_{1C} посылают для того, чтобы центральная ячейка поняла есть ли еще ячейки в состоянии L_1 или R_{1C} слева или справа. Этап сравнения длин закончится, когда не останется ни одной ячейки в состоянии L_1 или R_{1C}.

$t = 3$

q	S	E_E	L_0	L_1	L_0	L_1	L_0	$B_=$	$R_0\mathcal{C}$	$R_1\mathcal{C}$	$R_0\mathcal{C}$	$R_1\mathcal{C}$	$R_0\mathcal{C}$	$R_1\mathcal{C}$	$R_0\mathcal{C}$	$R_1\mathcal{C}$	$R_0\mathcal{C}$	E_E	S
ν_{-1}^1	0	0	0	1	0	1	0	1	1	0	1	0	1	0	1	0	1	0	0
ν_{-1}^2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ν_{-1}^3	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
ν_{-1}^4	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
ν_1^1	1	0	1	0	1	0	1	1	0	1	0	1	0	1	0	1	0	0	0
ν_1^2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ν_1^3	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0
ν_1^4	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0
ψ_1	0	0	0	1	0	1	0	0	0	1	0	1	0	1	0	1	0	0	0
ψ_2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ψ_3	0	0	0	1	0	1	0	0	0	1	0	1	0	1	0	1	0	0	0
ψ_4	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

В примере 2 центральная ячейка продолжает сравнивать сигналы, полученные по первой компоненте слева и справа.

$t = 4$

q	S	E_E	L_0	L_0	L_0	L_1	L_0	$B_=$	$R_0\mathcal{C}$	$R_1\mathcal{C}$	$R_0\mathcal{C}$	$R_0\mathcal{C}$	$R_0\mathcal{C}$	$R_1\mathcal{C}$	$R_0\mathcal{C}$	$R_0\mathcal{C}$	$R_0\mathcal{C}$	E_E	S
ν_{-1}^1	0	0	0	0	1	1	0	0	0	0	1	1	0	0	1	1	0	0	0
ν_{-1}^2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ν_{-1}^3	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
ν_{-1}^4	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
ν_1^1	0	0	0	1	1	0	0	0	0	1	1	0	0	1	1	0	0	0	0
ν_1^2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ν_1^3	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0
ν_1^4	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0
ψ_1	0	0	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	0	0
ψ_2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ψ_3	0	0	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	0	0
ψ_4	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

На шестом такте ячейка в состоянии $B_=$ слышит по первой компоненте слева 1, а справа 0, и переходит в состояние $B_>$.

$$\varphi(B_>, z_{\nu_{-1}}, z_{\nu_1}) = \begin{cases} B_<, \text{ если } z^1_{\nu_{-1}} = 0,\ z^1_{\nu_1} = 1; \\ B_R, \text{ если } z^3_{\nu_{-1}} = 1,\ z^3_{\nu_1} = 0; \end{cases}$$

$$\psi(B_>, z_{\nu_{-1}}, z_{\nu_1}) = \begin{cases} (0,0,0,3), \text{ если } z^3_{\nu_{-1}} = 1,\ z^3_{\nu_1} = 0; \\ (0,0,0,0), \text{ в остальных случаях.} \end{cases}$$

Переход центральной ячейки в состояние B_R будет означать, что длина отрицательного вектора больше.

$t = 5$

q	S	E_E	L_0	L_0	L_0	L_0	L_0	$B_>$	R_{0C}	R_{1C}	R_{0C}	R_{0C}	R_{0C}	R_{0C}	R_{0C}	R_{0C}	E_E	S
ν^1_{-1}	0	0	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1	1
ν^2_{-1}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ν^3_{-1}	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1
ν^4_{-1}	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
ν^1_1	1	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0	0	0
ν^2_1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ν^3_1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0
ν^4_1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0
ψ_1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
ψ_2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ψ_3	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
ψ_4	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

На седьмом такте ячейка в состоянии $B_>$ слышит по первой компоненте слева 0, а справа 1, и переходит в состояние $B_<$.

$$\varphi(B_<, z_{\nu_{-1}}, z_{\nu_1}) = \begin{cases} B_>, \text{ если } z^1_{\nu_{-1}} = 1,\ z^1_{\nu_1} = 0; \\ E_1, \text{ если } z^3_{\nu_{-1}} = 0,\ z^3_{\nu_1} = 1; \end{cases}$$

$$\psi(B_<, z_{\nu_{-1}}, z_{\nu_1}) = \begin{cases} (0,0,0,2), \text{ если } z^3_{\nu_{-1}} = 0,\ z^3_{\nu_1} = 1; \\ (0,0,0,0), \text{ в остальных случаях.} \end{cases}$$

Поскольку ячейка $B_<$ слышит по третьей компоненте слева 0, а справа 1, она понимает, что не осталось ни одной ячейки в состоянии L_1 и еще есть ячейки в состоянии R_{1C}. Значит отрицательный вектор короче. Ячейка $B_<$ посылает сигнал (0,0,0,2) и на следующем такте перейдет в состояние E_1.

$t = 6$

q	S	E_E	L_0	L_0	L_0	L_0	L_0	$B_<$	R_{0C}	R_{0C}	R_{0C}	R_{0C}	R_{0C}	R_{0C}	R_{0C}	R_{0C}	E_E	S
ν^1_{-1}	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
ν^2_{-1}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ν^3_{-1}	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
ν^4_{-1}	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
ν^1_1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0
ν^2_1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ν^3_1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0
ν^4_1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0
ψ_1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ψ_2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ψ_3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ψ_4	0	1	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	1	0

На восьмом такте ячейки L_0, услышав по четвертой компоненте справа 2, переходят в состояние L_1.

$$\varphi(L_0, z_{\nu_{-1}}, z_{\nu_1}) = \begin{cases} L_1, \text{ если } z^4_{\nu_1} = 2; \\ L_*, \text{ если } z^4_{\nu_1} = 3. \end{cases}$$

$$\psi(L_0, z_{\nu_{-1}}, z_{\nu_1}) = (0, 0, 0, 0).$$

Состояние L_* мы опишем позже.

Ячейки R_{1C} и R_{0C}, услышав по четвертой компоненте слева 2, переходят в состояние R_*. Состояние R_* — это однотактовый сон, ячейки в этом состоянии не посылают никаких сигналов и не реагируют на поступающие сигналы.

$$\varphi(R_{0C}, z_{\nu_{-1}}, z_{\nu_1}) = \begin{cases} R_{1C}, \text{ если } z^4_{\nu_{-1}} = 3; \\ R_*, \text{ если } z^4_{\nu_{-1}} = 2. \end{cases}$$

$$\psi(R_{0C}, z_{\nu_{-1}}, z_{\nu_1}) = (0, 0, 0, 0).$$

Ячейки в состоянии E_E переходят в состояние S.

$t = 7$

q	S	S	L_1	L_1	L_1	L_1	L_1	E_1	R_*	R_*	R_*	R_*	R_*	R_*	R_*	R_*	R_*	S	S
ν^1_{-1}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ν^2_{-1}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ν^3_{-1}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ν^4_{-1}	0	0	1	1	1	1	1	1	2	2	2	2	2	2	2	2	2	2	2
ν^1_1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ν^2_1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ν^3_1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ν^4_1	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	1	1	0
ψ_1	0	0	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0
ψ_2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ψ_3	0	0	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0
ψ_4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

На этом этап сравнения длин векторов заканчивается.

Этап 3. Вычисление суммы векторов.

Для примера 1 на третьем такте ячейка B_B переходит в состояние B_1. Состояние B_1 — это состояние, в которое переходит центральная ячейка, в случае когда оба вектора направлены положительно или отрицательно. Ближайшая к началу вектора ячейка E переходит в E_1, другая ячейка E переходит в C. Состояние C — это состояние ячеек, находящихся между началом и концом вектора, после завершения работы автомата. В этом состоянии ячейки ничего не посылают в эфир и не переходят в новые состояния.

$$\varphi(C, z_{\nu_{-1}}, z_{\nu_1}) = C.$$

$$\psi(C, z_{\nu_{-1}}, z_{\nu_1}) = (0, 0, 0, 0).$$

Ячейки L_+ переходят в активное состояние L_1, общее количество ячеек в состоянии L_1 вместе с ячейкой B_1 будет равно длине меньшего вектора.

$$\varphi(L_+, z_{\nu_{-1}}, z_{\nu_1}) = L_1.$$
$$\psi(L_+, z_{\nu_{-1}}, z_{\nu_1}) = (0, 0, 0, 0).$$

Ячейки R_C переходят в C, так как по окончании работы автомата они окажутся между началом и концом результирующего вектора.

$$\varphi(R_C, z_{\nu_{-1}}, z_{\nu_1}) = \begin{cases} C, \text{ если } z^3_{\nu_{-1}} = 1,\ z^4_{\nu_{-1}} = 2,\ z^3_{\nu_1} = 1,\ z^4_{\nu_1} = 0; \\ R_{1C}, \text{ в остальных случаях.} \end{cases}$$

$$\psi(R_C, z_{\nu_{-1}}, z_{\nu_1}) = (0, 0, 0, 0).$$

Ячейки R_- переходят в состояние S. Так как оба вектора положительно ориентированы, все "отрицательные" ячейки должны стать молчащими. В дальнейшем процессе они участвовать не будут.

$$\varphi(R_-, z_{\nu_{-1}}, z_{\nu_1}) = \begin{cases} R_{1-}, \text{ если } z^4_{\nu_1} = 0; \\ R_-, \text{ если } z^4_{\nu_1} = 4; \\ S, \text{ если } z^4_{\nu_1} = 2 \text{ или } z^4_{\nu_1} = 3. \end{cases}$$

$$\psi(R_-, z_{\nu_{-1}}, z_{\nu_1}) = (0, 0, 0, 0).$$

Состояние R_+ — это состояние двухтактового сна, на следующем такте, ячейки в этом состоянии перейдут в состояние R_1. Ячейкам в состоянии R_1 будет побитово передаваться двоичная запись длины меньшего вектора. Состояние R_1 должно появиться на один такт позже, чем L_1, так как ячейка E_1 должна сначала получить младший разряд в двоичной записи длины меньшего вектора и на следующем такте послать соответствующий сигнал.

$$\varphi(R_+, z_{\nu_{-1}}, z_{\nu_1}) = \begin{cases} R_1, \text{ если } z^4_{\nu_{-1}} = 0; \\ R_+, \text{ если } z^4_{\nu_{-1}} = 2; \\ S, \text{ если } z^4_{\nu_{-1}} = 3 \text{ или } z^4_{\nu_{-1}} = 4. \end{cases}$$

$$\psi(R_+, z_{\nu_{-1}}, z_{\nu_1}) = (0, 0, 0, 0).$$

$t = 2$

q	S	B_1	L_1	L_1	L_1	L_1	L_1	E_1	C	C	R_+	R_+	R_+	R_+	R_+	R_+	R_+	R_+
ν^1_{-1}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ν^2_{-1}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ν^3_{-1}	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1
ν^4_{-1}	0	0	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
ν^1_1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ν^2_1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ν^3_1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0
ν^4_1	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ψ_1	0	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0
ψ_2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ψ_3	0	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0
ψ_4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

На четвертом такте в примере 1 ячейка B_1 переходит в состояние сна B_*.

$$\varphi(B_1, z_{\nu_{-1}}, z_{\nu_1}) = B_*.$$
$$\psi(B_1, z_{\nu_{-1}}, z_{\nu_1}) = (1, 0, 1, 0).$$

Состояние B_* — это состояние, в котором центральная ячейка молчит, до тех пор пока не услышит сигнал о том, что не осталось ни одной ячейки в состоянии L_1. Это будет означать, что сумма векторов найдена. В случае получения соответствующего сигнала центральная ячейка станет началом результирующего вектора и перейдет в состояние L_2. Функции φ и ψ для состояния B_* мы опишем в разделе «Этап 4. Завершение работы».

Ячейки L_1 затираются, в данном случае переходят в состояние S, если по первой компоненте слева получили сигнал 0.

Ячейки R_+ переходят в активное состояние R_1, услышав слева сигнал (0,0,1,0). Количество ячеек в состоянии R_1 бесконечно, поскольку мы не можем заранее определить какая из ячеек в состоянии R_1 будет концом результирующего вектора.

Ячейка E_1 передает по четвертой компоненте сигнал, который она получила по первой компоненте слева. Если ячейка в состоянии E_1 получает по третьей компоненте слева 0, то она понимает, что не осталось ни одной ячейки в состоянии L_1 и переходит в состояние B_2. Состояние B_2 мы опишем позже.

$$\varphi(E_1, z_{\nu_{-1}}, z_{\nu_1}) = B_2, \text{ если } z^3_{\nu_{-1}} = 0.$$

$$\psi(E_1, z_{\nu_{-1}}, z_{\nu_1}) = \begin{cases} (0,0,0,0), \text{ если } z^1_{\nu_{-1}} = 0; \\ (0,0,0,1), \text{ если } z^1_{\nu_{-1}} = 1. \end{cases}$$

$t = 3$

q	S	B_*	L_1	S	L_1	S	L_1	E_1	C	C	R_1	R_1	R_1	R_1	R_1	R_1	R_1	R_1
ν^1_{-1}	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0
ν^2_{-1}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ν^3_{-1}	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
ν^4_{-1}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ν^1_1	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0
ν^2_1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ν^3_1	0	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0
ν^4_1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ψ_1	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0
ψ_2	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
ψ_3	0	0	1	0	1	0	1	0	0	0	2	2	2	2	2	2	2	2
ψ_4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

На следующих тактах, в примере 1, ячейка E_1 продолжает посылать по четвертой компоненте сигнал, который был получен по первой компоненте слева. Ячейки R_1 затираются (переходят в состояние S), если по

второй и четвертой компоненте слева слышат разные сигналы $\{0,1\}$. По окончании этапа определения длины результирующего вектора, самая левая ячейка R_1 перейдет в состояние R_2, остальные ячейки из состояния R_1 перейдут в состояние S.

$$\varphi(R_1, z_{\nu_{-1}}, z_{\nu_1}) = \begin{cases} R_1, \text{ если } z^2_{\nu_{-1}} = z^4_{\nu_{-1}}; \\ S, \text{ если } (z^2_{\nu_{-1}} = 0,\ z^4_{\nu_{-1}} = 1) \text{ или } (z^2_{\nu_{-1}} = 1,\ z^4_{\nu_{-1}} = 0); \\ S, \text{ если } z^3_{\nu_{-1}} = 2,\ z^4_{\nu_{-1}} = 2; \\ R_2, \text{ если } z^3_{\nu_{-1}} = 2,\ z^4_{\nu_{-1}} = 0. \end{cases}$$

$$\psi(R_1, z_{\nu_{-1}}, z_{\nu_1}) = (0,1,2,0).$$

Этап нахождения суммы векторов для примера 1 продолжается до тех пор пока не останется ни одной ячейки в состоянии L_1.

Автомат устроен так, что начиная с пятого такта мы не можем однозначно определить сигнал, который ячейки получают по второй компоненте справа, поскольку имеется бесконечное число ячеек в состоянии R_1.

$t = 4$

q	S	B_*	S	S	L_1	S	S	E_1	C	C	R_1	S	R_1	S	R_1	S	R_1	S
ν^1_{-1}	0	0	0	1	1	0	0	1	1	1	1	1	1	1	1	1	1	1
ν^2_{-1}	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	1	0	1
ν^3_{-1}	0	0	0	1	1	1	1	1	1	1	1	2	2	2	2	2	2	2
ν^4_{-1}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ν^1_1	1	1	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0
ν^2_1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ν^3_1	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
ν^4_1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ψ_1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
ψ_2	0	0	0	0	0	0	0	0	0	0	1	0	1	0	1	0	1	0
ψ_3	0	0	0	0	1	0	0	0	0	0	2	0	2	0	2	0	2	0
ψ_4	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0

$t = 5$

q	S	B_*	S	S	S	S	S	E_1	C	C	S	S	R_1	S	S	S	R_1	S
ν^1_{-1}	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1
ν^2_{-1}	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	1	1	0
ν^3_{-1}	0	0	0	0	0	1	1	1	1	1	1	2	2	2	2	2	2	2
ν^4_{-1}	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1
ν^1_1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ν^2_1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ν^3_1	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
ν^4_1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0
ψ_1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ψ_2	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0
ψ_3	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	2	0
ψ_4	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0

На последнем такте этапа вычисления суммы векторов ячейка E_1 не слышит сигнала по третьей компоненте слева и переходит в состояние

B_2. На этом этап вычисления суммы векторов для примера 1 завершается. В случае, когда оба вектора положительны, самая левая ячейка в состоянии R_1 будет концом результирующего вектора.

$t = 6$

q	S	B_*	S	S	S	S	S	B_2	C	C	S	S	S	S	S	S	R_1	S
ν^1_{-1}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ν^2_{-1}	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0
ν^3_{-1}	0	0	0	0	0	0	0	0	0	0	0	0	0	2	2	2	2	2
ν^4_{-1}	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1
ν^1_1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ν^2_1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ν^3_1	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
ν^4_1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0
ψ_1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ψ_2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
ψ_3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0
ψ_4	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0

Вернемся к примеру 2, т.е к случаю, когда вектора расположены по разные стороны от центральной точки. Ячейки, которые находятся в состоянии R_* перейдут в состояние R_R. Переход ячеек в состояние R_R говорит о том, что длина положительного вектора больше длины отрицательного вектора. Значит конец результирующего вектора будет справа от центральной ячейки. От длины положительного вектора мы будем отнимать длину отрицательного вектора. Ячейкам в состоянии R_R будет побитово передаваться двоичная запись длины отрицательного вектора. Состояние R_R должно появиться на один такт позже, чем L_1, так как ячейка E_1 должна сначала получить младший разряд в двоичной записи длины отрицательного вектора и на следующем такте послать соответствующий сигнал.

$$\varphi(R_*, z_{\nu_{-1}}, z_{\nu_1}) = R_R.$$
$$\psi(R_*, z_{\nu_{-1}}, z_{\nu_1}) = (0, 0, 0, 0).$$

Ячейка в состоянии E_1 передает по четвертой компоненте сигнал, который она получила по первой компоненте слева. То есть, центральная ячейка побитово передает двоичную запись длины отрицательного вектора. И, таким образом, ячейки, находящиеся в состоянии R_R, понимают которые из них должны перейти в состояние S.

$$\varphi(R_R, z_{\nu_{-1}}, z_{\nu_1}) = \begin{cases} R_R, \text{ если } z^1_{\nu_1} = z^4_{\nu_{-1}}; \\ S, \text{ если } (z^1_{\nu_1} = 0,\ z^4_{\nu_{-1}} = 1) \text{ или } (z^1_{\nu_1} = 1,\ z^4_{\nu_{-1}} = 0); \\ S, \text{ если } z^4_{\nu_{-1}} = 2,\ z^3_{\nu_1} = 2; \\ R_2, \text{ если } z^4_{\nu_{-1}} = 2,\ z^3_{\nu_1} = 0. \end{cases}$$

$$\psi(R_R, z_{\nu_{-1}}, z_{\nu_1}) = (1, 0, 2, 0).$$

Сигнал по третьей компоненте ячейки R_R посылают для того, чтобы, по завершении процесса нахождения суммы векторов, затереть все активные (за исключением центральной) ячейки слева от конца результирующего вектора. По завершении этапа вычисления суммы векторов самая правая ячейка в состоянии R_R будет концом результирующего вектора и перейдет в состояние R_2. Остальные ячейки в состоянии R_R затираются.

$t = 8$

q	S	S	S	L_1	S	L_1	S	E_1	R_R	R_R	R_R	R_R	R_R	R_R	R_R	R_R	S	S
ν^1_{-1}	0	0	0	1	0	1	0	1	1	1	1	1	1	1	1	1	1	1
ν^2_{-1}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ν^3_{-1}	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
ν^4_{-1}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ν^1_1	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0
ν^2_1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ν^3_1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0
ν^4_1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ψ_1	0	0	0	1	0	1	0	0	1	1	1	1	1	1	1	1	0	0
ψ_2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ψ_3	0	0	0	1	0	1	0	0	2	2	2	2	2	2	2	2	0	0
ψ_4	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0

$t = 9$

q	S	S	S	S	S	L_1	S	E_1	S	R_R	S	R_R	S	R_R	S	R_R	S	S	S
ν^1_{-1}	0	0	0	0	1	1	0	0	0	1	0	1	0	1	0	1	0	1	1
ν^2_{-1}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ν^3_{-1}	0	0	0	0	1	1	1	1	1	2	2	2	2	2	2	2	2	2	2
ν^4_{-1}	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1
ν^1_1	1	1	1	0	0	1	1	1	0	1	0	1	0	1	0	1	0	0	0
ν^2_1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ν^3_1	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	0	0	0
ν^4_1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0
ψ_1	0	0	0	0	0	1	0	0	0	1	0	1	0	1	0	1	0	0	0
ψ_2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ψ_3	0	0	0	0	0	1	0	0	0	2	0	2	0	2	0	2	0	0	0
ψ_4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

$t = 10$

q	S	S	S	S	S	S	S	E_1	S	S	S	R_R	S	S	S	R_R	S	S	S
ν^1_{-1}	0	0	0	0	0	0	1	1	1	1	0	0	1	1	0	0	1	1	1
ν^2_{-1}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ν^3_{-1}	0	0	0	0	0	0	1	1	1	1	2	2	2	2	2	2	2	2	2
ν^4_{-1}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ν^1_1	1	1	1	1	1	0	0	0	0	1	1	0	0	1	1	0	0	0	0
ν^2_1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ν^3_1	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	0	0	0	0
ν^4_1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ψ_1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0
ψ_2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ψ_3	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	2	0	0	0
ψ_4	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0

На следующем такте ячейка E_1 слышит по третьей компоненте слева 0, понимает, что не осталось ни одной ячейки в состоянии L_1 и переходит

в состояние B_2.

$$\varphi(B_2, z_{\nu_{-1}}, z_{\nu_1}) = \begin{cases} C, \text{ если } z^4_{\nu_{-1}} = z^4_{\nu_1} = 3; \\ L_2, \text{ если } z^3_{\nu_{-1}} = 2 \text{ или } z^3_{\nu_1} = 2. \end{cases}$$

$$\psi(B_2, z_{\nu_{-1}}, z_{\nu_1}) = (0, 0, 0, 2).$$

$t = 11$

q	S	S	S	S	S	S	S	B_2	S	S	S	S	S	S	S	R_R	S	S	S
ν^1_{-1}	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0
ν^2_{-1}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ν^3_{-1}	0	0	0	0	0	0	0	0	0	0	0	0	2	2	2	2	2	2	2
ν^4_{-1}	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1
ν^1_1	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0
ν^2_1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ν^3_1	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	0	0	0	0
ν^4_1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0
ψ_1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
ψ_2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ψ_3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0
ψ_4	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0

На этом этап нахождения суммы векторов для примера 2 завершается.

Этап 4: Завершение работы.

Для примера 1 на этом этапе все лишние ячейки в состоянии R_1 затираются, крайняя левая ячейка R_1 слышит слева сигнал (0,0,0,2) и переходит в состояние R_2. Ячейки R_1, которые надо затереть, определяем по третьей компоненте. Если ячейка R_1 слышит слева сигнал (0,0,2,2) или (0,1,2,2), то ячейка затирается, то есть переходит в состояние S. Ячейка в состоянии B_*, услышав сигнал от ячейки B_2, просыпается и переходит в состояние L_2.

$$\varphi(B_*, z_{\nu_{-1}}, z_{\nu_1}) = \begin{cases} L_2, \text{ если } z^4_{\nu_{-1}} = 2 \text{ или } z^4_{\nu_1} = 2; \\ B_*, \text{ в остальных случаях.} \end{cases}$$

$$\psi(B_*, z_{\nu_{-1}}, z_{\nu_1}) = (0, 0, 0, 0).$$

Ячейки в состоянии L_2 и R_2 посылают сигнал (0,0,0,3) для того, чтобы все ячейки между L_2 и R_2 перешли в состояние C.

$$\varphi(L_2, z_{\nu_{-1}}, z_{\nu_1}) = B_F.$$
$$\psi(L_2, z_{\nu_{-1}}, z_{\nu_1}) = (0, 0, 0, 3).$$

$$\varphi(R_2, z_{\nu_{-1}}, z_{\nu_1}) = E_F.$$
$$\psi(R_2, z_{\nu_{-1}}, z_{\nu_1}) = (0, 0, 0, 3).$$

$t = 7$

q	S	L_2	S	S	S	S	S	B_2	C	C	S	S	S	S	S	S	R_2	S
ν^1_{-1}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ν^2_{-1}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
ν^3_{-1}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ν^4_{-1}	0	0	0	0	0	0	0	0	2	2	2	2	2	2	2	2	2	2
ν^1_1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ν^2_1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ν^3_1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ν^4_1	2	2	2	2	2	2	2	0	0	0	0	0	0	0	0	0	0	0
ψ_1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ψ_2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ψ_3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ψ_4	0	3	0	0	0	0	0	2	0	0	0	0	0	0	0	0	3	0

Ячейки из состояния S могут перейти только в состояние покоя $*$ или в состояние C.

$$\varphi(S, z_{\nu_{-1}}, z_{\nu_1}) = \begin{cases} C, \text{ если } z^4_{\nu_{-1}} = z^4_{\nu_1} = 3; \\ *, \text{ если } (z^4_{\nu_{-1}} = 3,\ z^4_{\nu_1} = 0) \text{ или } (z^4_{\nu_{-1}} = 0,\ z^4_{\nu_1} = 3). \end{cases}$$

$$\psi(S, z_{\nu_{-1}}, z_{\nu_1}) = (0, 0, 0, 0).$$

На последнем такте ячейки L_2 и R_2 переходят в состояния B_F и E_F. Ячейка B_2 в примере 1 перейдет в состояние C.

$t = 8$

q	$*$	B_F	C	C	C	C	C	C	C	C	C	C	C	C	C	C	E_F	$*$
ν^1_{-1}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ν^2_{-1}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ν^3_{-1}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ν^4_{-1}	0	0	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
ν^1_1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ν^2_1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ν^3_1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ν^4_1	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	0	0
ψ_1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ψ_2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ψ_3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ψ_4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Состояния B_F и E_F — состояния ячеек начала и конца результирующего вектора, а C — это состояние ячеек, находящихся между началом и концом вектора, при завершении работы автомата. В этих состояниях ячейки ничего не посылают в эфир и не переходят в новые состояния. В примере 1 автомат завершает свою работу.

Для примера 2 самая правая ячейка в состоянии R_R слышит по четвертой компоненте слева 2, а по третьей справа 0 и переходит в состояние R_2. Остальные ячейки в состоянии R_R, если они есть, переходят в состояние S.

$t = 12$

q	S	S	S	S	S	S	S	L_2	S	S	S	S	S	S	S	R_2	S	S	S
ν_{-1}^1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1
ν_{-1}^2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ν_{-1}^3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	2	2
ν_{-1}^4	0	0	0	0	0	0	0	0	2	2	2	2	2	2	2	2	2	2	2
ν_1^1	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0
ν_1^2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ν_1^3	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	0	0	0	0
ν_1^4	2	2	2	2	2	2	2	0	0	0	0	0	0	0	0	0	0	0	0
ψ_1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ψ_2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ψ_3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ψ_4	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	3	0	0	0

На последнем такте ячейки в состоянии S, которые слышат слева и справа сигнал (0,0,0,3), переходят в состояние C. Остальные ячейки в состоянии S переходят в состояние $*$. Ячейка L_2 переходит в состояние B_F, ячейка R_2 — в состояние E_F. На этом автомат завершает свою работу.

$t = 13$

q	$*$	$*$	$*$	$*$	$*$	$*$	$*$	B_F	C	C	C	C	C	C	C	E_F	$*$	$*$	$*$
ν_{-1}^1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ν_{-1}^2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ν_{-1}^3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ν_{-1}^4	0	0	0	0	0	0	0	0	3	3	3	3	3	3	3	3	3	3	3
ν_1^1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ν_1^2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ν_1^3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ν_1^4	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	0	0	0	0
ψ_1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ψ_2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ψ_3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ψ_4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Описание функций φ и ψ для состояний в случае когда оба вектора отрицательные.

Принцип работы автомата в случае, когда оба вектора отрицательные, аналогичен описанному в примере 1. Дадим описание состояний для этого случая.

Состояние R_- — это состояние, которое принимают ячейки, находящиеся правее конца вектора большей длины. Это молчащие ячейки, которые переходят в состояние R_{1-} или в S в зависимости от полученных сигналов справа. Смысл состояния R_- тот же, что и состояния R_+.

$$\varphi(R_-, z_{\nu_{-1}}, z_{\nu_1}) = \begin{cases} R_{1-}, \text{ если } z_{\nu_1}^4 = 0; \\ R_-, \text{ если } z_{\nu_1}^4 = 4; \\ S, \text{ если } z_{\nu_1}^4 = 2 \text{ или } z_{\nu_1}^4 = 3. \end{cases}$$

$$\psi(R_-, z_{\nu_{-1}}, z_{\nu_1}) = (0, 0, 0, 0).$$

Состояние L_- — это состояние на втором такте для всех ячеек, находящихся между концом вектора меньшей длины и началом векторов. Смысл состояния L_- тот же, что и состояния L_+.

$$\varphi(L_-, z_{\nu_{-1}}, z_{\nu_1}) = L_{1-}.$$
$$\psi(L_-, z_{\nu_{-1}}, z_{\nu_1}) = (0,0,0,0).$$

Напомним, что в состояние L_C переходят ячейки, находящиеся между концами векторов, в случае когда оба вектора отрицательны, либо ячейки, находящиеся между началом и концом отрицательного вектора в случае, когда вектора направлены в противоположные стороны. В зависимости от сигнала, который посылают начало и концы векторов, ячейки в этом состоянии могут перейти в состояние C или L_1. В случае, когда оба вектора отрицательны L_C переходит в в состояние C.

$$\varphi(L_C, z_{\nu_{-1}}, z_{\nu_1}) = \begin{cases} C, \text{ если } z^3_{\nu_{-1}} = 1,\ z^4_{\nu_{-1}} = 0,\ z^3_{\nu_1} = 1,\ z^4_{\nu_1} = 4; \\ L_1, \text{ в остальных случаях.} \end{cases}$$

$$\psi(L_C, z_{\nu_{-1}}, z_{\nu_1}) = (0,0,0,0).$$

В состояние E_{1-} переходит конец вектора меньшей длины. Ячейка в этом состоянии побитово передает двоичную запись длины меньшего вектора по четвертой компоненте. Если ячейка в состоянии E_{1-} получает по третьей компоненте справа 0, то она понимает, что не осталось ни одной ячейки в состоянии L_{1-}, посылает сигнал (0,0,0,2) и переходит в состояние C.

$$\varphi(E_{1-}, z_{\nu_{-1}}, z_{\nu_1}) = C, \text{ если } z^3_{\nu_1} = 0.$$

$$\psi(E_{1-}, z_{\nu_{-1}}, z_{\nu_1}) = \begin{cases} (0,0,0,0), \text{ если } z^1_{\nu_1} = 0; \\ (0,0,0,1), \text{ если } z^1_{\nu_1} = 1; \\ (0,0,0,2), \text{ если } z^3_{\nu_1} = 0. \end{cases}$$

Состояние R_{1-} — это состояние в какой-то момент всех ячеек, находящихся правее конца вектора большей длины. Ячейки, находящиеся в этом состоянии, переходят в состояние S, если слышат разные сигналы 0,1 по второй и четвертой компоненте справа. По окончании этапа вычисления суммы векторов самая правая из ячеек в состоянии R_{1-} будет концом результирующего вектора и перейдет в состояние R_2. Остальные ячейки R_{1-} при завершении процесса затираются, то есть переходят в состояние S. Смысл состояния R_{1-} тот же, что и состояния R_1.

$$\varphi(R_{1-}, z_{\nu_{-1}}, z_{\nu_1}) = \begin{cases} R_{1-}, \text{ если } z^2_{\nu_1} = z^4_{\nu_1}; \\ S, \text{ если } (z^2_{\nu_1} = 0,\ z^4_{\nu_1} = 1) \text{ или } (z^2_{\nu_1} = 1,\ z^4_{\nu_1} = 0); \\ S, \text{ если } z^3_{\nu_1} = 2,\ z^4_{\nu_1} = 2; \\ R_2, \text{ если } z^3_{\nu_1} = 0,\ z^4_{\nu_1} = 2. \end{cases}$$

$$\psi(R_{1-}, z_{\nu_{-1}}, z_{\nu_1}) = (0,1,2,0).$$

Сигнал по третьей компоненте ячейки R_{1-} посылают для того, чтобы после завершения второго этапа затереть все активные ячейки справа от конца результирующего вектора.

Состояние L_{1-} — это состояние в какой-то момент всех ячеек, находящихся между началом и концом вектора меньшей длины. На каждом следующем такте каждая вторая ячейка, находящаяся в этом состоянии, переходит в состояние S. Смысл состояния L_{1-} тот же, что и состояния L_1.

$$\varphi(L_{1-}, z_{\nu_{-1}}, z_{\nu_1}) = \begin{cases} L_{1-}, \text{ если } z^1_{\nu_1} = 1; \\ S, \text{ если } z^1_{\nu_1} = 0. \end{cases}$$

$$\psi(L_{1-}, z_{\nu_{-1}}, z_{\nu_1}) = (1, 0, 1, 0).$$

Этап завершения работы, в случае когда оба вектора отрицательные, ничем не отличается от случая когда оба вектора положительные.

Описание функций φ и ψ для состояний в случае когда вектора направлены в противоположные стороны и длина отрицательного вектора больше.

В случае когда длина отрицательного вектора больше, чем длина положительного вектора принцип работы автомата аналогичен случаю описанному в примере 2.

Опишем состояния для этого случая.

В этом случае центральная ячейка переходит в состояние B_R из состояния $B_>$. Ячейка в состоянии B_R передает по четвертой компоненте сигнал, который она получила по первой компоненте справа.

$$\varphi(B_R, z_{\nu_{-1}}, z_{\nu_1}) = B_2, \text{ если } z^3_{\nu_1} = 0.$$

$$\psi(B_R, z_{\nu_{-1}}, z_{\nu_1}) = \begin{cases} (0, 0, 0, 0), \text{ если } z^1_{\nu_1} = 0; \\ (0, 0, 0, 1), \text{ если } z^1_{\nu_1} = 1; \end{cases}$$

Ячейки в состоянии L_1 и L_0 переходят в состояние L_*. Состояния L_* — это однотактовый сон, ячейки в этом состоянии не посылают никаких сигналов и не реагируют на поступающие сигналы. На следующем такте ячейки в этом состоянии перейдут в состояние L_L. Переход ячеек в состояние L_L говорит о том, что длина отрицательного вектора больше длины положительного вектора. Значит конец результирующего вектора будет слева от центральной ячейки. От длины отрицательного вектора мы будем отнимать длину положительного вектора. Ячейкам в состоянии L_L будет побитово передаваться двоичная запись длины положительного вектора. Состояние L_L должно появиться на один такт позже, чем R_{1C}, так как ячейка B_R должна сначала получить младший разряд в двоичной записи длины положительного вектора и на следующем такте послать соответствующий сигнал.

$$\varphi(L_*, z_{\nu_{-1}}, z_{\nu_1}) = L_L.$$
$$\psi(L_*, z_{\nu_{-1}}, z_{\nu_1}) = (0, 0, 0, 0).$$

Получив сигнал по четвертой компоненте, который передает ячейка B_R, ячейки в состоянии L_L понимают какие из них должны перейти в состояние S. По завершении этапа вычисления суммы векторов самая левая ячейка в состоянии L_L будет концом результирующего вектора и перейдет в состояние R_2. Остальные ячейки в состоянии L_L затираются (переходят в состояние S).

$$\varphi(L_L, z_{\nu_{-1}}, z_{\nu_1}) = \begin{cases} L_L, \text{ если } z^1_{\nu_{-1}} = z^4_{\nu_1}; \\ S, \text{ если } (z^1_{\nu_{-1}} = 0,\ z^4_{\nu_1} = 1) \text{ или } (z^1_{\nu_{-1}} = 1,\ z^4_{\nu_1} = 0); \\ S, \text{ если } z^3_{\nu_{-1}} = 2,\ z^4_{\nu_1} = 2; \\ R_2, \text{ если } z^3_{\nu_{-1}} = 0,\ z^4_{\nu_1} = 2. \end{cases}$$

$$\psi(L_L, z_{\nu_{-1}}, z_{\nu_1}) = (1, 0, 2, 0).$$

Сигнал по третьей компоненте ячейки L_L посылают для того, чтобы, после завершении этапа вычисления суммы векторов, затереть все активные (за исключением центральной) ячейки справа от конца результирующего вектора.

4. Время работы автомата

Максимальное время работы автомата $T = T_1 + T_2 + T_3 + T_4$ где T_i — количество тактов затраченное на выполнение соответствующего этапа.

T_1 равно константе 2.

T_2 это время выполнения этапа сравнения длин отрезков. В работе Васильева Д. И.[9] доказывается, что это время равно $\log_2 n + 2$, где n — длина меньшего отрезка.

T_3 равно $\log_2 n$, где n — длина меньшего вектора.

T_4 равно константе 2.

Таким образом $T \leq 2 \log_2 n + 6$.

Список литературы

[1] Дж. фон Нейман, *Теория самовоспроизводящихся автоматов*, Мир, Москва, 1971.

[2] Neumann J., von, *Collected works*, New York, 1961 – 1963.

[3] Neumann J., von, *Theory of self-reproducing automata*, London, 1966.

[4] Burks A., *Essays on Cellular Automata*, University of Illinois Press, 1971.

[5] Мур Э. Ф., “Математические модели самовоспроизведения”, *В кн.: Математические проблемы в биологии*, 1966.

[6] Кудрявцев В. Б., Подколзин А. С., Болотов А. А., *Основы теории однородных структур*, Наука, Москва, 1990.

[7] Гасанов Э. Э., “Клеточные автоматы с локаторами”, *Интеллектуальные системы. Теория и приложения*, **24**:2 (2020), 120–133.

[8] Калачев Г. В., “Замечания к определению клеточного автомата с локаторами”, *Интеллектуальные системы. Теория и приложения*, **24**:4 (2020), 121–133.

[9] Васильев Д. И., “Поиск юлижайшего соседа на прямой с помощью клеточного автомата с локаторами”, *Интеллектуальные системы. Теория и приложения*, **24**:3 (2020), 99–119.

The addition of vectors problem solution using the celluar automata with locators
Ibragimova D.E.

The article considers applying a cellular automaton with locators to solving the problem of vector addition. The locator cellular avtomaton model assumes the possibility for each cell to translate a signal through any distance. It is proven in this article that such possibility allows to decrease the problem complexity from linear to logarithmic (against the classic cellular automaton model).

Keywords: cellular automata, homogeneous structures, the problem of vector addition.

References

[1] Neumann J., von, *Theory of self-reproducing automata*, Мир, Moscow, 1971.

[2] Neumann J., von, *Collected works*, New York, 1961 – 1963.

[3] Neumann J., von, *Theory of self-reproducing automata*, London, 1966.

[4] Burks A., *Essays on Cellular Automata*, University of Illinois Press, 1971.

[5] Mur E. F., “Mathematical models of self-reproduction”, *In b.: Mathematical problems in biology*, 1966.

[6] Kudryavtsev V.B., Podkolzin A.S., Bolotov A.A., *Fundamentals of the theory of homogeneous structures*, «Science», Moscow, 1990.

[7] Gasanov E.E., “Cellular automata with locators”, *Intelligent systems. Theory and applications*, **24**:2 (2020), 120–133.

[8] Kalachev G.V., “Notes on the definition of a cellular automaton with locators”, *Intelligent systems. Theory and applications*, **24**:4 (2020), 121–133.

[9] Vasilev D.I., “The closest neighbour problem solution using the cellular automata with locators model”, *Intelligent systems. Theory and applications*, **24**:3 (2020), 99–119.

О сложности реализации элементарного базиса в классе одноместных линейных автоматов, сохраняющих нулевую последовательность.

И. Ю. Ильин[1]

Сейчас нам известны некоторые факты о полноте конечных множеств линейно-автоматных функций, найдены все предполные классы по операциям суперпозиции и композиции, выведены критерии полноты в терминах предполных классов. В данной работе доказаны оценки количества операций для выразимости задержки и нейтрального элемента в случая одноместных линейных автоматов, сохраняющих нулевую последовательность.

Ключевые слова: линейные автоматы, оценки сложности.

Необходимые для нашей работы определения можно найти в работах [1] и [2], однако, в целях упрощения, представляется правильным повторить их, чтобы исключить избыточные обращения к источникам.

Мы рассмотриваем множество рациональных дробей $E_2^{'}(\xi) = \{\frac{u}{v} | u, v \in E_2[\xi], v(0) = 1\}$ с операциями:

1. Сложения: $\mu_1, \mu_2 \in E_2^{'}(\xi) : \mu_1 + \mu_2$
2. Умножения: $\mu_1, \mu_2 \in E_2^{'}(\xi) : \mu_1 \mu_2$
3. Обратной связи: $\mu_1, \mu_2 \in E_2^{'}(\xi), \mu_2(0) = 0 : F_b(\mu_1, \mu_2) = \frac{\mu_1}{1+\mu_2}$.

Обозначим $K^{(1)}(M)$ - замыкание множества M по операциям сложения, умножения и обратной связи, а $S^{(1)}(M)$ - замыкание множества M только по операциям сложения и умножения.

Множество $M \subseteq E_2^{'}(\xi)$ - полно, если $K^{(1)}(M) = E_2^{'}(\xi)$

Назовем множество $\{1, \xi\}$ элементарным базисом в $E_2^{'}(\xi)$ по операциям $K^{(1)}$.

Введем последовательность p_i: $p_1 = \xi, p_2 = 1+\xi, ...$ состоящую из всех неприводимых многочленов.

Введем множества $R_i^{(1)}$ и $M_i^{(1)}$ следующим образом:

$R_0^{(1)} = \{\mu | \mu = \frac{u}{v}; u, v \in E_2[\xi], (u, v) = 1 : \deg(u) < \deg(v)\ \}$.

$R_i^{(1)} = \{\mu | \mu = \frac{u}{v}; u, v \in E_2[\xi], (u, v) = 1, up_i\}$.

$M_0^{(1)} = \{\mu | \mu = \frac{u}{v}; u, v \in E_2[\xi], (u, v) = 1 :$ если $u(0) = 0$, тогда $\deg(u) < \deg(v)$, а если $u(0) = 1$, тогда $\deg(u) = \deg(v)\ \}$.

[1] *Ильин Иван Юрьевич* — аспирант каф. математической теории интеллектуальных систем мех.-мат. ф-та МГУ, e-mail: vanyail@yandex.ru.

Ilin Ivan Yurievich— graduate student, Lomonosov Moscow State University, Faculty of Mechanics and Mathematics, Chair of Mathematical Theory of Intellectual Systems.

$M_i^{(1)} = \{\mu | \mu = \frac{u}{v}; u, v \in E_2[\xi], (u, v) = 1$ такие, что если $u(0) = 1 =>$ $1 + \frac{u}{v} = \frac{u+v}{v} = \frac{\xi p_i u'}{v}$, а если $u(0) = 0 => \frac{u}{v} = \frac{\xi p_i u'}{v}\}$.

$J^{(1)} = \{M_i^{(1)}, R_i^{(1)}, i = 0, 1, 2, ...\}$. В работе [1] показано, что множества $R_i^{(1)}$ и $M_i^{(1)}$ составляют приведенную критериальную систему предполных классов в $E_2'(\xi)$ относительно операций $K^{(1)}$.

Перед тем как сформулировать итоговую теорему, докажем вспомогательные леммы и утверждения.

Лемма 1. *Пусть* $K^{(1)}(M) = E_2'(\xi)$, *где* $M = \{\frac{u_i}{v_i} | u_i,\ v_i \in E_2[\xi],$ $v(0) = 1, i = 1, ...n\}, n \in \mathbb{N}$.

Обозначим $k = \max\{\deg(\mu_1), \deg(\mu_2), ..., \deg(\mu_n)\}$.

Тогда из M *можно получить рациональные дроби* $\mu, \mu\xi$, $\mu \neq 0$, *используя* $O(nk^2)$ *операций сложения и умножения,* $\deg(\mu) \leq 2^k k$.

Доказательство. Обозначим $F_i(z) = \mu_i(\xi)v_i(z) + u_i(z)$ - многочлены от z с коэффициентами из $E_2'(\xi)$. Каждый из этих многочленов строится за $O(1)$ операций умножения и сложения, таким образом, для построения всех многочленов мы потратим $O(n)$ операций сложения и умножения. Теперь найдем НОД этих многочленов.

Шаг 1. Возьмем $F_i(z)$ с минимальной степенью $\deg(F_i(z)) = p$. Уменьшим степень каждого $F_j(z), i \neq j$ на 1 по следующему правилу: Перепишем $F_j(z)$, вынесем старшую степень z^s вперед. $F_j(z) = z^s(\mu_j'(\xi)a_{j,s} + b_{j,s}) + ...$, где s - $\deg(F_j(z))$. $F_i(z) = z^p(\mu_i'(\xi)a_{i,k} + b_{i,k})$. Сократим сташую степень в $F_j(z)$: $F_j(z)$: $:= F_j(z)(\mu_i'(\xi)a_{i,k} + b_{i,k}) + z^{s-p}(\mu_j'(\xi)a_{j,s} + b_{j,s})F_i(z)$. Степень по ξ не выше чем $\deg(\mu_i') + \deg(\mu_i')$. Коэффициент перед z с наибольшей степенью переобозначим как μ_j'. Количество операций для получения всех коэффициентов многочлена F_j - $O(k)$, так как степень F_j не больше чем k. Поскольку мы проводим операцию уменьшения степени для всех F_j, то общее количество операций - $O(nk)$.

Шаг 2. Теперь будем повторять шаг 1 не более чем k раз, поскольку k - максимальная степень изначального $F_j(z)$.

Согласно работе [1] НОД многочленов $F_j(z)$ равен $\xi + z$. Таким образом не более чем за $O(nk^2)$ операций мы получим $\mu(\xi)(\xi + z)$, где μ - итоговый коэффициент перед НОД, который получаем в результате применения алгоритма, степень многочлена-коэффициента увеличивается не более чем в два раза от максимальной степени коэффициентов на предыдущем шаге, таким образом получаем оценку $2^k k$ на степень μ. □

Лемма 2. $M = \{\frac{u_i}{v_i} | u_i,\ v_i \in E_2[\xi],\ v(0) = 1, i = 1, ...n\}, n \in \mathbb{N}$.
$k = \max\{\deg(\mu_1), \deg(\mu_2), ..., \deg(\mu_n)\}$, $K^{(1)}(M) = E_2'(\xi)$. *Пусть по доказанной ранее лемме 1 были получены дроби* $\mu, \xi\mu$, *где* $\mu = \frac{u}{v}, \mu \neq 0$. *Тогда*

за $k\lceil \log k \rceil$ *операций сложения и умножения и 1 операцию обратной связи мы можем получить* $\xi\hat{\mu}, \hat{\mu}$, *где* $\hat{\mu} = \frac{\hat{u}}{\hat{v}}$ *такие, что* $\deg(\hat{u}) \geq \deg(\hat{v})$. *Степень итоговой дроби* $\hat{\mu}$ *не превосходит* $4k^2 2^k$.

Доказательство. В случае, если $deg(u) \geq deg(v)$, доказывать нечего. Предположим, что $\deg(u) < \deg(v)$. Так как M - полно, то существует дробь $\mu_R \in M \setminus R_0^{(1)}, \mu_R = \frac{u_R}{v_R}, \deg(u_R) \geq \deg(v_R), \deg(\mu_R) \leq k$.

Случай 1. Если $\deg(u_R) > \deg(v_R)$, то искомые дроби $\hat{\mu}, \xi\hat{\mu}$ мы можем получить за $\deg(\mu)$ операций. Напомним, что в предыдущей лемме мы доказали, что степень $\deg(\mu)$ не превосходит $k2^k$, тогда количество операций для получения самого $\hat{\mu}$ оценим как $k + \lceil \log k \rceil$, т.к. количество операций для получения $\mu_R^{k2^k}$ не превосходит $k + \lceil \log k \rceil$. Степень итоговой дроби $\hat{\mu}$ - не превосходит $2\deg(\mu_R)k2^k$.

Предположим теперь, что $\deg u_R = \deg v_R$.

Случай 2. $u_R(0) = 0$, то $\tilde{\mu} = Fb(\mu_R, \mu_R) = \frac{\mu_R}{1+\mu_R} = \mu_R \frac{v_R}{u_R + v_R}, \deg(v_R) > \deg(v_R + u_R) => \tilde{\mu} = \frac{\tilde{u_R}}{\tilde{v_R}}, \deg(\tilde{u}) > \deg(\tilde{v})$. Таким образом мы получили дробь, к которой можем применить доказательство из случая 1, $\deg(\tilde{\mu}) \leq 2\deg\mu_R$.

Случай 3. Если $u_R(0) = 1, \exists \mu_m \in M \setminus M_0^{(1)}, \mu_m = \frac{u_m}{v_m}$. Если $\deg(u_m) > \deg(v_m)$, то сводим к случаю 1. Если $\deg(u_m) = \deg(v_m)$ и $u_m(0) = 0$, то сводим к случаю 2. Если $\deg(u_m) < \deg(v_m) => u_m(0) = 1$. $\mu_R + \mu_m = \mu' = \frac{u'}{v'}$, $\deg(u') = \deg(v') : \mu_R + \mu_m = \frac{u_R v_m + v_R u_m}{v_R v_m}$. $\deg(u_R v_m + v_R u_m) = \deg(u_R v_m) = \deg(v_R u_m)$. $u'(0) = 1$, а значит имеем случай 2, $deg(\mu_R + \mu_m) \leq 2k$. По итогу мы используем не более 1 операций обратной связи и $k + \lceil \log k \rceil$ операций умножения и сложения. Степень итоговой дроби не превосходит $4k^2 2^k$. □

Лемма 3. $M = \{\frac{u_i}{v_i}, u_i,\ v_i \in E_2[\xi],\ v(0) = 1, i = 1, ...n\}, n \in \mathbb{N}$.
$k = \max\{\deg(\mu_1), \deg(\mu_2), ..., \deg(\mu_n)\}$, $K^{(1)}(M) = E_2'(\xi)$.
Пусть $\mu,\ \xi\mu \in K^{(1)}(M),\ \mu = \frac{u}{v} \neq 0, \deg(u) \geq \deg(v),\ \deg(\mu) \leq 4k^2 2^k$. *Тогда существует такое* $T \in \mathbb{N}$, *что* $\forall i, j \geq T : \xi^i \mu^j \in K^{(1)}(M)$, *дробь* $\xi^i \mu^j$ *может быть получена не более чем за* $O(max(i - j, 0)k^2 2^k + j)$ *операций сложения и умножения.*

Доказательство. Пусть $j \geq i$. Тогда перемножая $\xi\mu$ i-раз, мы получим $\xi^i \mu^i$. Домножим $\xi^i \mu^i$ на μ^{j-i} (перемноженное $j - i$ раз), получим $\xi^i \mu^j$, затрачивая не более j операций.

Теперь предположим, что $j < i$ и перепишем дробь μ следующим образом:

$$\mu = \frac{u}{v} = \frac{a_0 + a_1\xi + ...a_{p-1}\xi^{p-1} + \xi^p}{1 + b_1\xi + ...b_{k-1}\xi^{s-1} + \xi^s}, p \geq s, p \leq 4k^2 2^k.$$

Получим все степени μ^i до p, потратив на это $O(k^2 2^k)$ операций умножения, степень μ полагаем не превосходящей $4k^2 2^k$ (степень μ из прошлой леммы). Теперь формально домножим выражение $a_0 + a_1\xi + ...a_{p-1}\xi^{p-1} + \xi^p + \mu(1 + b_1\xi + ...b_{k-1}\xi^{s-1} + \xi^s) = 0$ на μ^{p-1}.

Получим $a_0\mu^{p-1} + a_1\xi\mu^{p-1} + ...a_{p-1}\xi^{p-1}\mu^{p-1} + \xi^p\mu^{p-1} + \mu^p(1 + b_1\xi + ...b_{k-1}\xi^{s-1} + \xi^s) = 0$, и перепишем в следующем виде:

$$a_0\mu^{p-1} + a_1\xi\mu^{p-1} + ...a_{p-1}\xi^{p-1}\mu^{p-1} + \mu^p(1 + b_1\xi + ...b_{k-1}\xi^{s-1} + \xi^s) = \xi^p\mu^{p-1}$$

Заметим теперь, что общее количество членов в левой части этого выражения не превышает $8k^2 2^k$, а общее количество операций для получения $\xi^p\mu^{p-1}$ - $O(k^2 2^k)$.

Теперь будем получать степени выше p+1: Снова запишем выражение

$$a_0 + a_1\xi + ...a_{p-1}\xi^{p-1} + \xi^p + \mu(1 + b_1\xi + ...b_{k-1}\xi^{s-1} + \xi^s) = 0$$

но на этот раз домножим его на $\mu^{p-1}\xi$ и получим выражение

$$a_0\xi\mu^{p-1} + a_1\xi^1\mu^{p-1} + ...a_{p-1}\xi^p\mu^{p-1} + \xi^{p+1}\mu^{p-1} + \mu^p(\xi + b_1\xi^2 + ...b_{k-1}\xi^s + \xi^{s+1}) = 0.$$

Перенесем $\xi^{p+1}\mu^{p-1}$ в правую часть и получим

$$a_0\xi\mu^{p-1} + a_1\xi^1\mu^{p-1} + ...a_{p-1}\xi^p\mu^{p-1} + \mu^p(\xi + b_1\xi^2 + ...b_{k-1}\xi^s + \xi^{s+1}) = \xi^{p+1}\mu^{p-1}.$$

На получение данного выражения потратим не более $O(k^2 2^k)$ операций.

Таким образом для получения дроби $\xi^i\mu^j$; где $i, j \geq 4k^2 2^k = T$ мы потратим не более $O(max(i - j, 0)k^2 2^k + j)$ операций.

□

Лемма 4. *$M = \{\frac{u_i}{v_i}, u_i,\ v_i \in E_2[\xi],\ v(0) = 1, i = 1, ...n\}, n \in \mathbb{N}$.*
$k = \max\{\deg(\mu_1), \deg(\mu_2), ..., \deg(\mu_n)\}$.

Пусть мы уже построили дробь $\xi^i\mu^s$, где $i, s = 4k^2 2^k$, тогда мы можем получить многочлены $\xi^j u_0$, где $j = i, ..., i + 256k^{10}2^{6k}$, $u_0 = a_0 + a_1\xi^1 + ... + a_T\xi^T$ не более чем за $O(k^9 2^{5k})$ операций, для T справедлива оценка $T \leq 16k^4 2^{2k}$.

Доказательство. Обозначим $\mu^s = \mu_0 = \frac{u_0}{v_0}$. Напомним, что $\deg(\mu) \leq (4k^2 2^k)$, поэтому $\deg(\mu_0) \leq 16k^4 2^{2k}$.

Выпишем многочлен $v_0 = b_0 + b_1\xi^1 + ... + b_T\xi^T$, $T \leq 16k^4 2^{2k}$.

По предыдущей лемме получим дроби $\xi^i\mu_0, \xi^{i+1}\mu_0, ..., \xi^{T+i}, ...,$ $\xi^{i+16k^5 2^{3k}}\mu_0, ..., \xi^{i+256k^{10}2^{6k}+T}\mu_0$ и посчитаем необходимое количество операций. Имея дробь $\xi^l\mu_0$ мы можем получить дробь $\xi^{l+1}\mu_0$ за $O(k^2 2^k)$ операций, а значит, получение всех таких дробей до $\xi^{i+256k^{10}2^{6k}+T}\mu_0$ будет иметь сложность $O(k^{11}2^{7k})$.

Затем каждую дробь $\xi^j\mu_0$ домножим на соответствующие константы b_l и просуммируем с $\xi^j\mu_0$ с:

$$\xi^j\mu_0 + b_1\xi^{j+1}\mu_0 + b_2\xi^{j+2}\mu_0 + ... + b_T\xi^{j+T}\mu_0 =$$

$$= \xi^j\mu_0(1 + b_1\xi^1 b_2\xi^2 + ... + b_T\xi^T) =$$

$$= \xi^j\mu_0 v_0 = \xi^j u_0,$$

$$\text{где } u_0 = p_{i_1}^{j_1} p_{i_2}^{j_2} ... p_{i_s}^{j_s}$$

.

Количество используемых операций сложения и умножения для этого суммирования не превышает $\deg(v_0)$ или $O(k^4 2^{2k})$. Так как мы проделываем этот процесс для всех j от i до $256k^{10}2^{6k}$, то итоговая оценка на количество операций сложения и умножения будет $O(k^{14}2^{10k})$. □

Лемма 5. *$M = \{\frac{u_i}{v_i}, u_i,\ v_i \in E_2[\xi],\ v(0) = 1, i = 1, ...n\}, n \in \mathbb{N}$.*
$k = \max\{\deg(\mu_1), \deg(\mu_2), ..., \deg(\mu_n)\}$. M не содержится в θ $\forall\theta, \theta \in J^{(1)}$, следовательно, $\forall i, i \in \{2, 3, ...\}\exists\mu_i \in K^{(1)}(M), \mu_i = \frac{u_i}{v_i}, (u_i, v_i) = 1, v_i p_i$. Количество операций сложения и умножения для получения μ_i - $O(k)$, количество операций обратной связи для получения μ_i не превышает одной.

Доказательство. Случай 1. Так как M целиком не содержится в предполных классах, то существует $\mu \in M$, $\mu \notin M_i^{(1)}, R_i^{(1)}$. $\mu = \frac{u}{v}$, p_i не делит u. Если p_i делит v, то доказывать нечего. Предположим, что p_i не делит v.

Случай 1а. $\mu = \frac{\xi u'}{v}, p_i$ не делит u'.

$1 + \mu^T = \frac{v^T + (\xi u')^T}{v^T}$, существует такой $T \in \mathbb{N}$, что p_i делит $v^T + (\xi u')^T$.

Пусть $\deg(p_i) = m$, тогда $v^{2^m-1} + (\xi u')^{2^m-1} \equiv 1(\mod p_i)$. Получить μ^{2^m-1} мы можем не более чем за $2m$ операций: будем получать последовательно степени μ: μ^2, μ^4, μ^8... (это займет не более m операций), а затем перемножим нужные нам степени (это также займет не больше m операций).

Теперь, применяя обратную связь к дробям $F_b(\mu, \mu^T)$ получим выражение $\frac{\mu}{1+\mu^T} = \frac{u}{v}\frac{v^T}{v^T+(\xi u')^T}$,где знаменатель делится на p_i.

Заметим, что $m \leq k$, поэтому мы получаем оценку $O(k)$ на количество операций сложения и умножения.

Случай 1б. Если $\mu(0) = 1$, то рассмотрим дробь

$$\mu' = \mu + \mu^2 = \mu(1+\mu) = \frac{u}{v}(\frac{u+v}{v}), \mu(0) = 0, \mu'(0) = 1$$

.

$\mu \notin R_i^{(1)} => p_i$ не делит u, $\mu \in M_i^{(1)} => p_i$ не делит $u+v$

p_i не делит $u(u+v)$, а значит мы свели случай 1б к случаю 1а за 1 операцию сложения и одну операцию умножения.

Случай 2.

$\exists \mu_1 \in M \setminus M_i^{(1)}, \exists \mu_2 \in M \setminus R_i^{(1)}$.

$\mu_1 \in R_i^{(1)}, \mu_2 \in M_i^{(1)}$.

$\mu_1 + \mu_2 = \mu, \mu \notin R_i^{(1)}, \mu \notin M_i^{(1)}$, иначе бы $\mu + \mu_1 = \mu_2$ принадлежал бы $R_i^{(1)}$. $\deg(\mu)$ в данном случае не превышает $k2^k$. Таким образом за одну операцию сложения мы сводим случай 2 к случаю 1.

По итогу, для получения требуемой дроби мы тратим не более 1 операции обратной связи и $O(k)$ операций сложения и умножения.

□

Лемма 6. *$M = \{\frac{u_i}{v_i}, u_i, v_i \in E_2[\xi], v(0) = 1, i = 1, ...n\}, n \in \mathbb{N}$.*
$k = \max\{\deg(\mu_1), \deg(\mu_2), ..., \deg(\mu_n)\}$. Если существует многочлены $\xi^i u'$,
$...\xi^{i+256k^{10}2^{6k}}u'$, такие, что степень $\deg(u') \leq 16k^4 2^{2k}$, а сам многочлен u' имеет вид $u' = p_{i_2}^{j_2} \cdot p_{i_3}^{j_3} ... \cdot p_{i_l}^{j_l}$ и такое натуральное число i_1, что $\forall i \geq i_1$ выполнено: $\xi^i u' \in K^{(1)}(M)$, то количество операций для получения ξ^i не превышает $O(k^{18}2^{10k})$.

Доказательство. Пусть у нас есть многочлен $\xi^i u', i \geq j_1, ..., u' = p_{i_2}^{j_2} \cdot ...p_{i_l}^{j_l}$, количество операций для получения этого многочлена посчитано ранее, в леммах 2, 3.

По лемме 5 мы можем получить дробь $\mu_2 = \frac{u_2}{v_2}, (u_2, v_2) = 1, v_2 p_{i_2}$ за $O(k)$ операций. Из леммы 5 видно, что степень μ_2 не превосходит $k2^k$.

Домножим $\xi^i u'$ на $\mu_2^{j_2}$ и получим дробь $\xi^i \frac{u_2'}{v_2'} \in K^{(1)}(M)$, где p_{i_2} не делит u_2', $i \in \mathrm{N}$, $i \geq j_1$, j_2 не превосходит $\deg(u')$, а $\deg(u')$ в свою очередь не превосходит $16k^4 2^{2k}$. Следовательно, степень дроби $\frac{u_2'}{v_2'}$ не превосходит $16k^5 2^{3k}$. Из дробей $\xi^i \frac{u_2'}{v_2'}, ..., \xi^{i+16k^5 2^{3k}} \frac{u_2'}{v_2'}$, используя метод из леммы 4, мы можем получить многочлен $\xi^i u_2'$. Количество операций не будет превышать $\deg(v_2')$, а степень u_2' по итогу не будет превышать

$16k^5 2^{3k}$. Помимо многочлена $\xi^i u'_2$ нам необходимо получить степени $\xi^j u'_2$, $j = i, ..., i + 256k^{10}2^{6k}$, это потребуется в дальшейшем доказательстве. Сделать это мы можем за $O(k^{14}2^{8k})$ операций.

Аналогичным образом получим многочлены $u'_t, t = 3, .., l$, где каждый из многочленов не делится на p_{i_t} и потратим на это не более $O(lk^{14}2^{8k})$ операций.

НОД$(u', u'_2, ..., u'_l)$=1, а значит существуют многочлены $v_1, ..., v_l \in E_2[\xi]$ такие, что $v_1 u' + v_2 u'_2 + ... + v_l u'_l = 1$.

Обозначим $v_i(j)$ коэффициент перед многочленом u'_i на j-й итерации расширенного алгоритма Евклида, который мы будем применять ниже. На первом шаге которого найдем расширенный

$$\textbf{НОД}(u'\xi^i, u'_2\xi^i) = R(1)\xi^i$$
$$v_1(1)u'\xi^i + v_2(1)u'_2\xi^i = R(1)\xi^i, p(\xi, 1) = v_1(1),$$

т.е. $p(\xi, j)$ - многочлен перед первым слагаемым в результате работы расширенного алгоритма Евклида. Далее получаем

$$\textbf{НОД}(R(1)\xi^i, u'_3\xi^i) = R(2)\xi^i$$
$$p(\xi, 2)R(1)\xi^i + v_3(2)u'_3\xi^i = R(2)\xi^i$$
$$v_1(2)u'\xi^i + v_2(2)u'_2\xi^i + v_3(2)u'_3\xi^i = R(2)\xi^i$$
$$v_1(2) = v_1(1)p(\xi, 1), v_2(2) = v_2(1)p(\xi, 1)$$
$$\deg(p(\xi, j)) + deg(v_i(j)) \leq (j + 1)(16k^5 2^{3k}).$$

Видно, что итоговая степень v_i не превосходит $256k^1 02^{10k}$, и эти необходимые нам степени многочленов $\xi^j u'_s$, $j = i, ..., i + 256k^{10}2^{6k}, s = 1, ..., l$ были получены нами ранее. На последней итерации алгоритма, таким образом, будем иметь нужное нам многочлен:

$$v_1 u'\xi^i + v_2 u'_2\xi^i + ... + v_l u'_l\xi^i = \xi^i.$$

Вычисление каждого из итоговых $v_i(l)$ будет занимать не более $O(k^{10}2^{6k})$ операций. А итоговое количество операций ссложения и умножения для получения v_i будет не более $O(l(k^{10}2^{5k})$. Поскольку $l \leq (16k^4 2^{2k})$, то при уже построенных многочленах $\xi^j u'_s$, $j = i, ..., i + 256k^{10}2^{6k}, s = 1, ..., l$ сложность получения **НОД** не превосходит $O((k^{12}2^{8k}))$. Сложность построения этих многочленов мы оценивалии ранее как $O(lk^{14}2^{8k})$. Подставляя в эту оценку $l = (16k^4 2^{2k})$, получаем итоговую сложность получения ξ и в $O(k^{18}2^{10k})$.

□

Лемма 7. *$M = \{\frac{u_i}{v_i}, u_i,\ v_i \in E_2[\xi],\ v(0) = 1, i = 1, ...n\}, n \in \mathbb{N}$.*
$k = \max\{\deg(\mu_1), \deg(\mu_2), ..., \deg(\mu_n)\}$.
Если $M \not\subseteq \theta\ \forall \theta \in J^{(1)}$, то $\forall m \in \mathbb{N}, \forall u \in E_2[\xi] : \deg(u) < m, \exists \mu \in E_2'(\xi)$, что $u + \xi^m\mu \in K^{(1)}(M)$, где $\xi^m\mu = a_0\xi^m + a_1\xi^{m+1} + a_2\xi^{m+2} + ...$ Дробь $u + \xi^m\mu$ может быть получена за $O(m)$ операций.

Доказательство.

$$\mu \in M \setminus R_1^{(1)} => \mu(0) = 1,$$

$$\mu = 1 + a_1\xi + a_2\xi^2 + ... + a_i\xi^i + ...$$

$$\mu^2 = 1 + a_1\xi^2 + a_2\xi^4 + ... + a_i\xi^{2i} + ...$$

$$\mu^{2^\tau} = 1 + a_1\xi^{2^\tau} + a_2\xi^{2\cdot 2^\tau} + ... = 1 + \xi^{2^\tau}\mu'.$$

Мы можем выбрать $\tau = \lceil \log_2 m \rceil$ и таким образом для $u = 1$ получаем, что $1 + \xi^{\lceil \log 2m}\mu_1 \in K^1(M)$, применяя $\lceil \log_2 m \rceil$ операций умножения.

Выберем $\mu_1' \in M \setminus M_1^{(1)} => \mu_1' + \mu_1'(0) = \xi + b_2\xi^2 + ...$

$$\hat{\mu_1} = \begin{cases} \mu_1', & \text{если } \mu_1(0) = 0 \\ \mu_1' + (1 + \xi^n\mu_1) & \text{если } \mu_1(0) = 1 \end{cases}$$

Для $m = 1$ мы уже все доказали, докажем для $m > 1$.
$\hat{\mu} = \xi + c_2^1\xi^2 + ...$
$\hat{\mu}_s = (\hat{\mu})^s, s = 1, ..., m-1.$
$\hat{\mu}_s = \xi^s + c_{s+1}^s\xi^{s+1} + ...$
$\hat{\mu}_s = \xi^s + \xi^{s+1}\tilde{\mu}_s.$
Эти дроби мы получаем используя $O(m)$ операций.
$\xi + \xi^m\tilde{\mu} \in K^{(1)}(\{\xi^k + \xi^{k+1}\tilde{\mu} | k = 1, ..m-1\})$. Многочлен $\xi + \xi^m$ может быть получен за $O(m)$ операций.

Доказательство. Будем доказывать по индукции.
При $m = 2$ все доказано. Докажем переход индукции $m - 1 \to m$.
$\xi + \xi^{m-1}\tilde{\tilde{\mu}}_m \in K^{(1)}(M)$ по предположению индукции.
Если $\tilde{\tilde{\mu}}_m(0) = 0$, то $\tilde{\tilde{\mu}}_m = \xi\mu_m''$, а значит $\xi + \xi\mu_m'' \in K^{(1)}(M)$.
Если $\tilde{\tilde{\mu}}_m(0) = 1$, то $\xi + \xi^{m-1}\tilde{\tilde{\mu}}_m + (\xi^{m-1} + \xi^m\tilde{\mu}_{m-1}) = \xi + \xi^m\mu^*$. □

Итак, нам необходимо получить $u + \xi^m\mu$, где $\deg(u) < m, \mu \in E_2'(\xi)$.
$u = a_0 + a_1\xi + a_2\xi^2 + ...a_{m-1}\xi^{m-1}$.
Мы можем получить следующее выражение: $a_0(1 + \xi^m\mu_0^*) + a_1(\xi + \xi^m\mu_1^*) + a_2(\xi + \xi^m\mu_2^*) + a_{m-1}(\xi^{m-1} + \xi^m\mu_{m-1}^*) = a_0 + a_1\xi + a_2\xi^2 + ...a_{m-1}\xi^{m-1} + \xi^m(\mu_0^* + \mu_1^* + ... + \mu_{m-1}^*) = u + \xi^m\mu^*$.
Всего на получение $u + \xi^m\mu^*$ мы потратили не более $O(m)$ операций.
□

Теорема 1. *Пусть у нас есть множество* $M = \{\frac{u_i}{v_i}, u_i,\ v_i \in E_2[\xi], v(0) = 1, i = 1, ...n\}, n \in \mathbb{N}$. $k = \max\{\deg(\mu_1), \deg(\mu_2), ..., \deg(\mu_n)\}$. $K^{(1)}(M) = E_2'(\xi)$. *Тогда для того, чтобы из* M *получить элементарный базис* $\{1, \xi\}$ *нам необходимо не более* $O(k^{18}2^{10k}+nk^2)$ *операций сложения и умножения, а также не более 7 операций обратной связи.*

Доказательство. Для $i = 16k^4 2^{2k}$ по лемме 6 мы можем получить $\xi^i \in K^{(1)}(M)$ за $O(k^{18}2^{10k}+nk^2)$ операций сложения и умножения, количество операций обратной связи - не более 4.

Запишем произвольную дробь вида $\mu = u + \xi^i \mu'$ из $E_2'(\xi)$, где $u \in E_2[\xi], \deg(u) < i, \mu' \in E_2'(\xi)$.

По лемме 7 мы можем получить $u + \xi^i \mu''$ за $O(i)$ операций. И теперь перезапишем $\mu = (u + \xi^T \mu'') + (\xi^T \mu'' + \xi^T \mu')$.

$\xi^T \mu'' + \xi^T = \xi^T(\mu'' + \mu') = \frac{\xi^T u}{v}$.

$v \in E_2'[\xi], v(0) = 1 => \exists s \in \mathbb{N}, \exists v' \in E_2'[\xi], vv' = 1 + \xi^s$.

По модулю v не более чем $2^{16k^4 2^{2k}}$ остатков, а значит этот остаток v' мы можем получить не более чем за $O(k^4 2^{2k})$ операций.

$(1 + \xi^s)^{2^\tau} = 1 + \xi^{s2^\tau}$.

$v'(1 + \xi^s)^{2^{\tau+1}} => vv'(1 + \xi^s)^{2^{\tau+1}} = 1 + \xi^{s2^\tau}$. Выберем τ таким, чтобы $\tau \leq i$.

$\frac{\xi^i u}{v} = \frac{\xi^i uv'}{vv'} = \frac{\xi^i uv'}{1+\xi^s} = Fb(\xi^T uv, \xi^s) => \mu \in K^{(1)}(M)$.

Тем же самым способом построим дробь $\mu + \xi$. Складываем μ и $\mu + \xi$ и получаем многочлен ξ. Аналогично, мы можем получить дробь $\mu + 1$, а затем получить единицу. Для получения μ, $\mu + \xi$, $\mu + 1$ нам потребуется не более трех операций обратной связи.

Общее количество операций сложения и умножения составляет $O(k^{12}2^{6k}+nk^2)$, а общее количество операций обратной связи - не больше 7.

□

Заключение.

Таким образом получены оценки на получение простейших функций из множества рациональных дробей над полем E_2. Дальнейшее направление данной работы представляется нам довольно очевидным: вывод оценок для полных подмножеств множества линейно-автоматных функций, проверка работоспособности выведенных соотношений для дробей над произвольными числовыми полями.

Автор выражает благодарность своему научному руководителю А.А. Часовских.

Список литературы

[1] Часовских А. А., “О полноте в классе линейных автоматов”, *Математические вопросы кибернетики*, 1991, №2, 140–166.

[2] Kudryavtsev V.B., Alyoshin S.V., Podkolzin A.S., *Introduction to automata theory*, «Science», Moscow, 1985, 320 с.

Comlexity of implementation of elementary basis in one-place lineary automata class that preserves zero sequence

Ilin I.Y.

For now some facts about completeness of linear automata are proven. We now know about every precomlete class on superposition operetion and composition operation, the completeness criterions are also had been formulated. In this work we have proven some facts about the complexity of this process: to receive neutral element and delay.

Keywords: linear automata, comlexity estimation.

References

[1] Часовских А. А., “О полноте в классе линейных автоматов”, *Математические вопросы кибернетики*, 1991, №2, 140–166

[2] Kudryavtsev V.B., Alyoshin S.V., Podkolzin A.S., *Introduction to automata theory*, «Science», Moscow, 1985, 320 с.

Восстановление выпуклых функций класса CPL нейронными сетями над ReLU-базисами

В. Г. Шишляков[1]

В работе рассматривается вопрос о классах функций, получаемых при использовании нейронных сетей над базисами с нелинейностями типа max. Сначала в работе рассматриваются некоторые свойства непрерывных кусочно-линейных функций и порождающих их классов эквивалентности. Затем, на базе этих свойств доказывается теорема о том, что нейронные сети, построенные над базисом, состоящим из всех линейных функций и максимумов от любого числа аргументов в качестве нелинейностей, могут в точности восстанавливать любую выпуклую непрерывную кусочно-линейную функцию.

Затем в работе рассматривается переход к RELU-базису, который является частным случаем базисов с нелинейностями типа max и доказывается теорема, аналогичная теореме, упомянутой выше. Также в работе обсуждается вопрос об оценке количества нейронов и слоев в полученных архитектурах.

Доказательство всех упомянутых теорем конструктивно, то есть в них явно строятся архитектуры нейронных сетей, удовлетворяющих вышеописанным свойствам.

Ключевые слова: Нейронные сети, архитектура, восстановление функций, выразимость функций, выпуклые функции, кусочно-линейные функции, функция ReLU, функция максимума.

1. Введение

Вопросы выразимости функций нейронными сетями начали изучаться в работе [1]. В указанном исследовании были впервые сформулированы и использованы некоторые определения, связывающие понятия нейронных сетей с классическими понятиями дискретной математики, такие, как схемы функциональных элементов и автоматные функции [2].

Базовой идеей в [1], которая используется и в данном исследовании, являлось рассмотрение нейронной сети с точки зрения схем функциональных элементов над определенным заранее фиксированным на-

[1] *Шишляков Владимир Геннадьевич* — аспирант каф. математической теории интеллектуальных систем мех.-мат. ф-та МГУ, e-mail: bolotmaks@yandex.ru.

Shishlyakov Vladimir Gennad'evich — graduate student, Lomonosov Moscow State University, Faculty of Mechanics and Mathematics, Chair of Mathematical Theory of Intellectual Systems.

бором функциональных элементов (базисом) и сравнение (с теоретико-множественной точки зрения) множества функций, соответствующих всем таким схемам, с классом функций, обладающих определенными свойствами.

Упомянутые выше схемы называются нейронными схемами и, по сути, являются синонимами термина нейронная сеть (нейронную сеть можно получить из нейронной схемы, группируя определенные комбинации функциональных элементов в нейроны, что подробно описано в работах [3] и [4]).

Также в [1] были впервые введены две числовые характеристики схем, именуемые нелинейной сложностью и глубиной, которые являются примерными оценками числа нейронов и числа слоев, соответственно, в нейронной сети, к которой можно преобразовать рассматриваемую нейронную схему.

В работе [1] было показано, что класс функций, реализуемых классическими нейронными сетями Мак-Каллока и Питтса (с функциями активации Хевисайда) [5] в точности совпадает с классом кусочно-параллельных функций. Однако, данный класс функций является достаточно узким классом, при помощи которого невозможно полноценно решать, например, задачи регрессии.

Есть и другая классическая функция активации, которая часто используется в современных архитектурах – это функция $RELU$. Возникает теоретический интерес – выяснить, какой класс функций выражается при помощи нейронных сетей с указанной функцией активации.

Учитывая, что базис для построения нейронных сетей с $RELU$ функциями активации содержит лишь кусочно-линейные непрерывные функции, а классы кусочно-линейных и непрерывных функций замкнуты по операции суперпозиции [1],[6], то возникает гипотеза, что, скорее всего, класс функций, представимых нейронными сетями над базисами с функциями активации $RELU$, совпадает с классом кусочно-линейных непрерывных функций.

Данная гипотеза частично была исследована в работе [6]. В указанной работе было доказано, что нейронные сети с $RELU$ функциями активации могут восстанавливать произвольную кусочно-линейную непрерывную функцию от одной и двух переменных. Но вопрос исследования функций с бóльшим числом переменных в работе [6] затронут не был.

Итоговый результат мною уже доказан, но будет публиковаться частями из-за большого объема материала. Данная статья открывает серию таких публикаций. В рассматриваемой работе приводится промежуточный, но очень важный результат, в котором показывается, что любая выпуклая (и, соответственно, вогнутая) кусочно-линейная непрерывная

функция может быть представлена в виде нейронной сети с функциями активации типа $RELU$.

При доказательстве данного результата был исследован более общий базис, состоящий из линейных функций и единственного класса нелинейностей – $\max(x_1, ..., x_n)$, а основной результат был доказан уже, как следствие результата, полученного над общим базисом.

Также в работе оценивается нелинейная сложность и нелинейная глубина полученных нейронных схем.

2. Основные понятия

Для начала определим основные понятия, которые используются в данной статье.

Нейронные сети можно рассматривать с двух точек зрения. С одной стороны, на них можно смотреть, как на функции с большим количеством подбираемых параметров (весов), а с другой стороны – как на схемы, реализующие эти функции. Поэтому при рассмотрении искусственных нейронных сетей со схематической точки зрения естественным образом возникает понятие базиса нейронной сети. Введем понятие базиса и связанные с ним понятия, следуя работам [1], [2] и [3].

Определение 1. *Базисом будем называть некоторый набор функциональных элементов, где каждый функциональный элемент представляет из себя пару $(S, f(x_1, ..., x_n))$, в которой $f(x_1, ..., x_n) : \mathbb{R}^n \to \mathbb{R}$, а S – сопоставленный ей графический объект с n входными стрелками и одной выходной (кратко – входы и выход объекта S). Входам объекта S приписаны слева направо переменные $x_1, ..., x_n$, являющиеся аргументами функции f, выходу приписан выход функции f.*

В данной работе будем рассматривать два базиса (1) и (2), приведенных ниже.

$$B_1 = \{c, \gamma \cdot x, \sum_n (x_1, ..., x_n), \max(x_1, ..., x_n)\} \tag{1}$$

$$B_2 = \{c, \gamma \cdot x, \sum_n (x_1, ..., x_n), RELU(x)\} \tag{2}$$

В базисах (1) и (2) используются следующие классы функций:

- Сумматор — каждая функция данного класса суммирует определенное количество входных аргументов и обозначается $\sum_n (x_1, ..., x_n)$.

- Максимум — каждая функция данного класса выбирает максимум из n аргументов $x_1, ..., x_n$.

- Константа — каждая функция данного класса выдает константу (у данной функции нет входных аргументов, каждый раз, когда на схему приходят входные данные, константа выдает одинаковое, заранее определенное, значение). Предполагается, что для каждого действительного числа c есть свой элемент-константа c из данного класса.

- Усилитель (умножение на константу) — в данном классе каждая функция умножает пришедший на вход аргумент x на фиксированную константу γ. Предполагается, что для каждого действительного числа γ есть свой элемент-усилитель $\gamma \cdot x$ из данного класса.

- RELU — определяется следующим образом: $RELU(x) = \max(x, 0)$.

Графические изображения данных элементов приведены на рис. 1 (а, б, в, г, д).

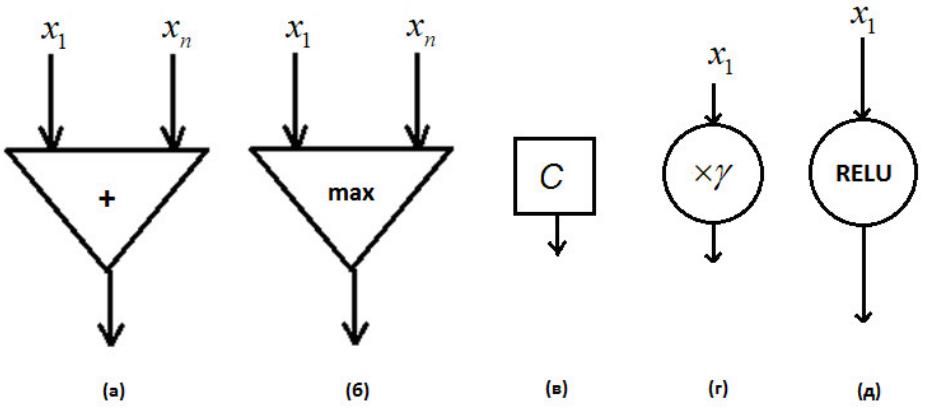


Рис. 1. Функциональные элементы рассматриваемых базисов

Функциональные элементы max и $RELU$ из базисов (1) и (2) будем называть нелинейными. Все остальные элементы данных базисов будем называть линейными.

Подробнее о том, как строятся схемы функциональных элементов из элементов любого из рассматриваемых в данной работе базисов, описано в [1]. Получаемые схемы называются нейронными схемами. Множество

нейронных схем, получаемых из элементов некоторого базиса B, будем обозначать $[B]$.

Будем говорить, что функция $f : \mathbb{R}^n \to \mathbb{R}$ представима (выражается, восстанавливается) нейронной схемой над некоторым базисом B, если $\exists g \in [B] : \forall \overline{x} \in \mathbb{R}^n \; f(\overline{x}) = g(\overline{x})$.

Определение 2. *Путем в нейронной схеме назовем последовательность функциональных элементов $(S_1, f_1), ..., (S_k, f_k)$ схемы, таких, что:*

- *Один из входов элемента (S_1, f_1) является входом схемы*
- *Выход элемента (S_k, f_k) является выходом схемы*
- *Один из входов элемента (S_i, f_i) является выходом элемента (S_{i-1}, f_{i-1}) при $i \in \{2, ..., k\}$*

Определение 3. *Длиной пути называется число нелинейных элементов, содержащихся в нем.*

Определение 4. *Нелинейной глубиной схемы называется длина самого длинного пути в данной схеме. Нелинейной сложностью схемы называется число всех нелинейных элементов в схеме.*

Далее введем важнейшие понятия для данной работы — это понятия гиперплоскости, класса эквивалентности, кусочно-линейной функции, функции класса CPL, а также связанные с ними понятия.

Определение 5. *Пусть дано пространство $\mathbb{R}^n$. Гиперплоскостью (плоскостью размерности $n-1$) будем называть геометрическое место точек $l = \{(x_1, ..., x_n) \in \mathbb{R}^n | a_1x_1 + ... + a_nx_n + c = 0\}$ (где $a_1, ..., a_n, c$ – константы, причем $a_1, ..., a_n$ не равны нулю одновременно).*

Пусть $\overline{x} = (x_1, ..., x_n) \in \mathbb{R}^n$. Тогда под значением $l(\overline{x})$ будем понимать значение $a_1x_1+...+a_nx_n+c$. Будем говорить, что точка $\overline{x}$ лежит на гиперплоскости l, если $l(\overline{x}) = 0$. Таким образом, гиперплоскость – это множество решений уравнения $l(\overline{x}) = 0$.

Так как по уравнению вида $a_1x_1 + ... + a_nx_n = 0$ можно в точности восстановить гиперплоскость, то в дальнейшем будем иногда определять гиперплоскость через ее уравнение, а само уравнение будем называть уравнением, определяющим гиперплоскость.

Вместе с гиперплоскостью $l = \{(x_1, ..., x_n) \in \mathbb{R}^n | a_1x_1 + ... + a_nx_n + c = 0\}$ естественным образом определяются два полупространства $l^+ = \{(x_1, ..., x_n) \in \mathbb{R}^n | a_1x_1 + ... + a_nx_n + c > 0\}$ и $l^- = \{(x_1, ..., x_n) \in \mathbb{R}^n | a_1x_1 + ... + a_nx_n + c < 0\}$. Очевидно, что $l^+ \cup l^- \cup l = \mathbb{R}^n$ для любой гиперплоскости l.

Определение 6. *Гиперплоскостью размерности $n-k$ будем называть геометрическое место точек, полученное, как пересечение k гиперплоскостей размерности $n-1$.*

Определение 7. *Пусть $\overline{x} = (x_1, ..., x_n) \in \mathbb{R}^n$, а $l_1, ..., l_k$ – некоторые гиперплоскости, определяемые выражениями $l_i = \{(x_1, ..., x_n) \in \mathbb{R}^n | a_{i1}x_1 + ... + a_{in}x_n + c_i = 0\}$, $i = 1, ..., k$.*

Тогда все пространство $\mathbb{R}^n$ разбивается гиперплоскостями $l_1, ..., l_k$ на классы эквивалентности. Рассмотрим функцию $\sigma(\overline{x}) = (sgn(a_{11}x_1 + ... + a_{1n}x_n + c_1), ..., sgn(a_{k1}x_1 + ... + a_{kn}x_n + c_k))$, определяющую – в каком куске пространства лежит точка $\overline{x} \in \mathbb{R}^n$.

Вектор-функция $\sigma(\overline{x})$ называется сигнатурой вектора $\overline{x}$ [1]. Каждая ее компонента показывает расположение точки $\overline{x}$ относительно каждой из гиперплоскостей $l_1, ..., , l_k$.

Определение 8. *Будем считать точки $\overline{x}, \overline{y} \in \mathbb{R}^n$ эквивалентными относительно гиперплоскостей $l_1, ..., l_k$ и обозначать это $\overline{x} \sim \overline{y}$, если $\sigma(\overline{x}) = \sigma(\overline{y})$.*

Очевидно, что отношение $\overline{x} \sim \overline{y}$ является отношением эквивалентности [1]. Следовательно, гиперплоскости разбивают $\mathbb{R}^n$ на множество непересекающихся классов $R^1, ..., R^s$, дающих в объединении все множество $\mathbb{R}^n$.

Определение 9. *Множество классов эквивалентности $\{R^1, ..., R^s\}$, полученное разбиением пространства $\mathbb{R}^n$ некоторым набором гиперплоскостей $\{l_1, ..., l_k\}$ при помощи введенного выше отношения эквивалентности $\overline{x} \sim \overline{y}$, будем называть разбиением пространства $\mathbb{R}^n$ или просто разбиением.*

Определение 10. *Сигнатурой класса $R^i \in \{R^1, ..., R^s\}$ назовем значение $\sigma(R^i) = \sigma(\overline{x})$, где $\overline{x}$ – произвольный элемент класса R^i. Отметим, что определение корректно, так как по построению $\forall \overline{x}, \overline{y} \in R^i \; \sigma(\overline{x}) = \sigma(\overline{y})$ [1].*

Определение 11. *Класс эквивалентности R^i назовем плоским, если $\exists l_j \in \{l_1, ..., l_k\} : R^i \subset l_j$.*

Определение 12. *Класс эквивалентности R^i назовем объемным, если он не является плоским.*

Очевидно, что если R^i – непустой и объемный, то он имеет сигнатуру $\sigma = (\sigma_1, ..., \sigma_k) : \sigma_j \neq 0 \; \forall j \in \{1, ..., k\}$. Если же класс R^i – непустой и плоский, то он имеет сигнатуру $\sigma = (\sigma_1, ..., \sigma_k)$, в которой $\exists j \in \{1, ..., k\} : \sigma_j = 0$.

Определение 13. *Пусть дан объемный класс эквивалентности* $R^i \subset \mathbb{R}^n$*, имеющий сигнатуру* $\sigma = (\sigma_1, ..., \sigma_k)$*. Тогда контуром размерности* $n - l, l \in \{1, ..., n\}$ *данного класса назовем множество всех классов эквивалентности, имеющих сигнатуры* σ'*, содержащиеся во множестве векторов, полученных из сигнатуры* $\sigma = (\sigma_1, ..., \sigma_k)$ *занулением всех возможных комбинаций из* l *компонент вектора* σ*.*

Контуром будем называть объединение контуров размерностей $n - 1, n - 2, ..., 0$*. Контур класса* R^i *обозначим* $\partial_{lin}\ R^i$*. Контур класса* R^i *размерности* k *будем обозначать* $\partial^k_{lin}\ R^i, k \in \{0, ..., n - 1\}$*.*

Определение 14. *Пусть пространство* $\mathbb{R}^n$ *разбивается на классы* $R^1, ..., R^s$ *гиперплоскостями* $l_1, ..., l_k$*. Будем говорить, что* $f(\overline{x})$ *является кусочно-линейной [1] и обозначать это* $f(\overline{x}) \in PL$*, если* $\forall R^i \in \{R^1, ..., R^s\}$ *верно, что* $\forall \overline{x} \in R^i\ f(\overline{x}) = \overline{b_i} \cdot \overline{x} + d_i$*, где* $\overline{b_i}, d_i$ *– константы.*

Всякую функцию $f_{R^i}(\overline{x}) = \overline{b_i} \cdot \overline{x} + d_i$ *будем называть линейной частью функции* $f(\overline{x})$ *на классе* R^i*.*

Иногда, чтобы сразу ввести кусочно-линейную функцию и разбиение пространства $\mathbb{R}^n$, над которым она задана, будем говорить, что функция $f(\overline{x})$ - кусочно-линейная функция, заданная над (разбиением, классами) $\{R^1, ..., R^s\}$.

Определение 15. *CPL-функцией назовем всякую кусочно-линейную непрерывную функцию. Множество всех непрерывных кусочно-линейных функций обозначим* CPL*.*

Наконец, напомним несколько определений и свойств из топологии и выпуклого анализа, которые будут активно использоваться в дальнейшем.

Определение 16. *Отрезком, соединяющим две произвольные точки* $\overline{x}, \overline{y} \in \mathbb{R}^n$ *назовем множество* $[\overline{x}; \overline{y}] = \{\overline{\xi} = (1 - \alpha) \cdot \overline{x} + \alpha \cdot \overline{y} \mid \alpha \in [0; 1]\}$*. Аналогично вводится понятие интервала:* $(\overline{x}; \overline{y}) = \{\overline{\xi} = (1 - \alpha) \cdot \overline{x} + \alpha \cdot \overline{y} \mid \alpha \in (0; 1)\}$*.*

Определение 17. *Множество* $M \subset \mathbb{R}^n$ *называется выпуклым, если выполняется, что* $\forall \overline{x}, \overline{y} \in M\ [\overline{x}; \overline{y}] \subset M$*.*

Определение 18. *Функция* $f : U \to \mathbb{R}$ *называется выпуклой, если* $U \subset \mathbb{R}^n$ *– выпуклое множество и* $\forall \overline{x_1}, \overline{x_2} \in U\ \forall \alpha \in (0; 1)$ *выполняется, что* $f((1 - \alpha) \cdot \overline{x_1} + \alpha \cdot \overline{x_2}) \leqslant (1 - \alpha) \cdot f(\overline{x_1}) + \alpha \cdot f(\overline{x_2})$*.*

Определение 19. *ε-окрестностью точки* $\overline{x} \in \mathbb{R}^n$ *($\varepsilon > 0$) будем называть шар* $O_\varepsilon(\overline{x}) = \{\overline{x'} \in \mathbb{R}^n : ||\overline{x} - \overline{x'}|| < \varepsilon\}$*. Иногда удобно рассуждать в терминах окрестностей вида* $O(\overline{x})$*. Под такой окрестностью понимается окрестность какого-то радиуса* $\varepsilon > 0$*, без уточнения конкретного*

значения ε. Обычно такая запись употребляется вместе с кванторами всеобщности или существования, когда конкретное значение ε становится избыточным.

Точка $\overline{x} \in M \subset \mathbb{R}^n$ называется внутренней точкой множества M, если $\exists O(\overline{x}) \subset M$. Множество всех внутренних точек множества M называется внутренностью множества M и обозначается $\operatorname{int} M$.

Определение 20. *Множество $M \subset \mathbb{R}^n$ называется открытым, если $\forall \overline{x} \in M \ \exists O(\overline{x}) : O(\overline{x}) \subset M$. Или, другими словами, если любая точка данного множества – внутренняя.*

Определение 21. *Точкой прикосновения множества $M \subset \mathbb{R}^n$ называется всякая точка $\overline{x'} \in \mathbb{R}^n$ для которой выполняется, что $\exists \{\overline{x_n}\}_{n\in\mathbb{N}} \subset M : \lim\limits_{n\to+\infty} \overline{x_n} = \overline{x'}$.*

Замыканием множества M назовем множество $\overline{M}$ всех точек прикосновения множества M.

Определение 22. *Граничной точкой множества $M \subset \mathbb{R}^n$ называется всякая точка $\overline{x} \in \mathbb{R}^n$, для которой выполняется, что $\forall O(\overline{x}) \ \exists \overline{x'}, \overline{x''} \in O(\overline{x}) : \overline{x'} \notin M \ \& \ \overline{x''} \in M$. Множество всех граничных точек множества M называется границей множества M и обозначается ∂M.*

Из курса общей топологии [7], [8] известно, что если множество M – открыто, то это эквивалентно тому, что $M = \operatorname{int} M$, а также, что $\overline{M} = \operatorname{int} M \cup \partial M$ и для любых множеств $A_1, ..., A_n$ верно, что $\overline{A_1 \cap ... \cap A_n} \subset \overline{A_1} \cap ... \cap \overline{A_n}$.

3. Основные результаты

Далее представлены основные результаты данной работы. Сначала представлены несколько вспомогательных утверждений (технических лемм), в которых изучаются различные свойства гиперплоскостей, классов эквивалентности, линейных и кусочно-линейных функций, необходимых для доказательства основного результата. Затем доказывается основное утверждение о том, что любую выпуклую CPL-функцию можно выразить нейронной схемой над базисом B_1. Наконец, доказывается следствие, в котором основное утверждение переносится на базис B_2.

Лемма 1. *Пусть $f(\overline{x}) = f(x_1, ..., x_n) = a_1 \cdot x_1 + ... + a_n \cdot x_n + d$ – линейная функция и имеются две точки $\overline{x}, \overline{y} \in \mathbb{R}^n$. Тогда выполняется $\forall \alpha \in [0; 1]$ $f((1-\alpha) \cdot \overline{x} + \alpha \cdot \overline{y}) = (1-\alpha) \cdot f(\overline{x}) + \alpha \cdot f(\overline{y})$.*

Доказательство. Пусть $\overline{x} = (x_1, ..., x_n)$ и $\overline{y} = (y_1, ..., y_n)$. Рассмотрим выражение $(1-\alpha) \cdot f(\overline{x}) + \alpha \cdot f(\overline{y})$ и распишем f по определению. Получим

$(1-\alpha)\cdot(a_1\cdot x_1+...+a_n\cdot x_n+d)+\alpha\cdot(a_1\cdot y_1+...+a_n\cdot y_n+d)$. Но тогда, раскрывая в последнем выражении скобки и приводя подобные, получаем выражение (3).

$$((1-\alpha)\cdot a_1\cdot x_1+\alpha\cdot a_1\cdot y_1)+... \\ +((1-\alpha)\cdot a_n\cdot x_n+\alpha\cdot a_n\cdot y_n)+((1-\alpha)\cdot d+\alpha\cdot d) \quad (3)$$

Наконец, учитывая, что $(1-\alpha)\cdot d+\alpha\cdot d=d$, выражение (3) можно записать в виде $a_1\cdot((1-\alpha)\cdot x_1+\alpha\cdot y_1)+...+a_n\cdot((1-\alpha)\cdot x_n+\alpha\cdot y_n)+d$. Последнее выражение по определению равно значению $f((1-\alpha)\cdot\overline{x}+\alpha\cdot\overline{y})$, что и доказывает данное утверждение. □

Лемма 2. *Пусть l – гиперплоскость, определяемая уравнением $a_1\cdot x_1+...+a_n\cdot x_n+d=0$, а также l^+ и l^- – соответствующие полупространства, определяемые данной гиперплоскостью. Тогда верны следующие утверждения:*

1) $l=\partial l^+, l=\partial l^-$

2) l^+, l^- – открытые множества

3) $\overline{l^+}=l^+\cup l$

4) $l^+, l^-, \overline{l^+}, \overline{l^-}, l$ – выпуклые множества

Доказательство. 1. Покажем, что $l=\partial l^+$.

Докажем включение $l\subset\partial l^+$.

Возьмем $\forall\overline{x}=(x_1,...,x_n)\in l$. Тогда для взятого $\overline{x}$, по определению l, выполняется равенство $a_1\cdot x_1+...+a_n\cdot x_n+d=0$. Затем возьмем $\forall\varepsilon>0$ и рассмотрим $O_\varepsilon(\overline{x})$. Так как, по определению, $l^+=\{(x_1,...,x_n)|a_1\cdot x_1+...+a_n\cdot x_n+d>0\}$, то $\overline{x}\notin l^+$.

Теперь обозначим $\overline{a}=(a_1,...,a_n)\neq\overline{0}$, где $\overline{0}=(0,...,0)\in\mathbb{R}^n$ и определим точку $\overline{x'}=\overline{x}+\frac{\varepsilon}{2\cdot||\overline{a}||}\cdot\overline{a}$, где $||\overline{a}||=\sqrt{a_1^2+...+a_n^2}$.

Отметим, что $\overline{x'}\in l^+$. Действительно, рассмотрим значение $l(\overline{x}+\frac{\varepsilon}{2\cdot||\overline{a}||}\cdot\overline{a})$. По определению функции $l(\overline{x})$, данное значение представимо в следующем виде:

$$a_1\cdot(x_1+\frac{\varepsilon}{2\cdot||\overline{a}||}\cdot a_1)+...+a_n\cdot(x_n+\frac{\varepsilon}{2\cdot||\overline{a}||}\cdot a_n)+d \quad (4)$$

Раскрывая в (4) скобки и группируя слагаемые, получаем выражение (5).

$$a_1\cdot x_1+...+a_n\cdot x_n+d+a_1^2\cdot\frac{\varepsilon}{2\cdot||\overline{a}||}+...+a_n^2\cdot\frac{\varepsilon}{2\cdot||\overline{a}||} \quad (5)$$

Заменяя в (5) значение $a_1 \cdot x_1 + ... + a_n \cdot x_n + d$ на $l(\overline{x})$ и $a_1^2 + ... + a_n^2$ на $||\overline{a}||^2$, получаем выражение (6).

$$l(\overline{x}) + ||\overline{a}||^2 \frac{\varepsilon}{2 \cdot ||\overline{a}||} \tag{6}$$

Причем $l(\overline{x}) = 0$, так как по условию $\overline{x} \in l$. Поэтому полученное выражение (6) можно преобразовать следующим образом:

$$l(\overline{x}) + ||\overline{a}||^2 \frac{\varepsilon}{2 \cdot ||\overline{a}||} = 0 + ||\overline{a}||^2 \frac{\varepsilon}{2 \cdot ||\overline{a}||} = \frac{\varepsilon \cdot ||\overline{a}||}{2} \tag{7}$$

Но $\frac{\varepsilon \cdot ||\overline{a}||}{2} > 0$, откуда, учитывая равенство выражений $l(\overline{x} + \frac{\varepsilon}{2 \cdot ||\overline{a}||} \cdot \overline{a})$ и (4), (5), (6), (7), получаем, что $l(\overline{x} + \frac{\varepsilon}{2 \cdot ||\overline{a}||} \cdot \overline{a}) > 0$. А последнее означает, по определению, что $\overline{x'} \in l^+$.

При этом, из определения $\overline{x'}$ следует, что $||\overline{x'} - \overline{x}|| = \|\overline{x} + \frac{\varepsilon \cdot \overline{a}}{2 \cdot ||\overline{a}||} - \overline{x}\| = \|\frac{\varepsilon \cdot \overline{a}}{2 \cdot ||\overline{a}||}\| = \frac{\varepsilon}{2 \cdot ||\overline{a}||} \cdot \|\overline{a}\| = \frac{\varepsilon}{2} < \varepsilon$, откуда следует, что $\overline{x'} \in O_\varepsilon(\overline{x})$.

Таким образом, имеем, что $\exists \overline{x'} \in O_\varepsilon(\overline{x}) \cap l^+$ и $\exists \overline{x} \in O_\varepsilon(\overline{x}) \cap (\mathbb{R}^n \setminus l^+)$. В силу произвольности выбора $\varepsilon > 0$ получаем, что в любой окрестности точки $\overline{x} \in l$ есть как точки из l^+, так и точки не из l^+. Последнее означает, по определению, что $\overline{x} \in \partial l^+$. Откуда, в силу выбора $\overline{x}$, немедленно вытекает доказываемое включение $l \subset \partial l^+$.

Осталось показать обратное включение $\partial l^+ \subset l$.

Возьмем $\forall \overline{x} \in \partial l^+$. Тогда по определению ∂l^+ и учитывая, что выражения $\overline{x'} \in l^+, \overline{x''} \notin l^+$ эквивалентны выражениям $l(\overline{x'}) > 0, l(\overline{x''}) \leqslant 0$, соответственно, верно следующее:

$$\forall \varepsilon > 0 \ \exists \overline{x'}, \overline{x''} \in O_\varepsilon(\overline{x}) : \ l(\overline{x'}) > 0, l(\overline{x''}) \leqslant 0 \tag{8}$$

Далее рассмотрим последовательность $\varepsilon_n = \frac{1}{n}, n = 1, 2, ...$ Тогда из (8) следуют выражения (9) и (10).

$$\exists \{\overline{x'_n}\}_{n \in \mathbb{N}} : \overline{x'_n} \in O_{\frac{1}{n}}(\overline{x}) \& l(\overline{x'_n}) > 0 \tag{9}$$

$$\exists \{\overline{x''_n}\}_{n \in \mathbb{N}} : \overline{x''_n} \in O_{\frac{1}{n}}(\overline{x}) \& l(\overline{x''_n}) \leqslant 0 \tag{10}$$

Из (9) и (10) с одной стороны следует, что $\{\overline{x'_n}\} \underset{n \to \infty}{\to} \overline{x}$ и $\{\overline{x''_n}\} \underset{n \to \infty}{\to} \overline{x}$, а с другой стороны верно, что $\forall n \in \mathbb{N} \ \ l(\overline{x'_n}) > 0$ и $l(\overline{x''_n}) \leqslant 0$.

Но тогда по теореме о предельном переходе в неравенстве получаем, что $l(\overline{x}) \geqslant 0$ и $l(\overline{x}) \leqslant 0$ одновременно. Следовательно, $l(\overline{x}) = 0$.

Итак, взяв $\forall \overline{x} \in \partial l^+$, получили, что $l(\overline{x}) = 0$, то есть, что $\overline{x} \in l$. Следовательно, $\partial l^+ \subset l$.

Из доказанных включений вытекает, что $l = \partial l^+$. Полностью аналогично доказывается, что $l = \partial l^-$.

2. Покажем, что l^+ открыто. Рассмотрим $\forall \overline{x} \in l^+$. Последнее означает, что $l(\overline{x}) > 0$. Возьмем также $\varepsilon = \frac{l(\overline{x})}{2 \cdot n^2 \cdot ||\overline{a}||} > 0$, где $\overline{a} = (a_1, ..., a_n)$ и $||\overline{a}||$ имеют тот же смысл, что и при доказательстве свойства 1.

Но тогда, воспользовавшись неравенством Минковского, а затем неравенством Гёльдера, получаем, что $\forall \overline{x'} \in O_\varepsilon(\overline{x})$ выражение $|l(\overline{x'}) - l(\overline{x})|$ имеет следующую оценку сверху:

$$\begin{gathered}
|l(\overline{x'}) - l(\overline{x})| = |a_1 \cdot (x'_1 - x_1) + ... + a_n \cdot (x'_n - x_n)| \leqslant \\
|a_1| \cdot |x'_1 - x_1| + ... + |a_n| \cdot |x'_n - x_n| \leqslant \\
|a_1| \cdot (|x'_1 - x_1| + ... + |x'_n - x_n|) + ... + |a_n| \cdot (|x'_1 - x_1| + ... + |x'_n - x_n|) = \\
(|a_1| + ... + |a_n|) \cdot (|x'_1 - x_1| + ... + |x'_n - x_n|) \leqslant \\
n \cdot ||\overline{a}|| \cdot n \cdot ||\overline{x'} - \overline{x}|| = n^2 \cdot ||\overline{a}|| \cdot ||\overline{x'} - \overline{x}|| < n^2 \cdot ||\overline{a}|| \cdot \varepsilon = \\
n^2 \cdot ||\overline{a}|| \cdot \frac{l(\overline{x})}{2 \cdot n^2 \cdot ||\overline{a}||} = \frac{l(\overline{x})}{2} \quad (11)
\end{gathered}$$

Раскрывая в полученной оценке (11) модуль $|l(\overline{x'}) - l(\overline{x})|$, получаем следующее выражение:

$$-\frac{l(\overline{x})}{2} < l(\overline{x'}) - l(\overline{x}) < \frac{l(\overline{x})}{2} \quad (12)$$

Но тогда, используя нижнюю оценку в (12) и тот факт, что $l(\overline{x}) > 0$, получаем следующее:

$$l(\overline{x'}) = l(\overline{x'}) + l(\overline{x}) - l(\overline{x}) = l(\overline{x}) + (l(\overline{x'}) - l(\overline{x})) > l(\overline{x}) - \frac{l(\overline{x})}{2} = \frac{l(\overline{x})}{2} > 0 \quad (13)$$

Из (13) следует, что $l(\overline{x'}) > 0$, а последнее означает, что $\overline{x'} \in l^+$.

Таким образом, показано, что для любой точки $\overline{x} \in l^+$ найдется некоторая окрестность $O_\varepsilon(\overline{x})$ такая, что $\forall \overline{x'} \in O_\varepsilon(\overline{x})$ верно $\overline{x'} \in l^+$. Из сказанного немедленно вытекает открытость множества l^+.

Аналогично доказывается, что l^- – открыто.

3. Известным фактом [7] является следующее выражение:

$$\overline{l^+} = \text{int } l^+ \cup \partial l^+ \quad (14)$$

Учитывая, что, по пункту 1 данной леммы, $\partial l^+ = l$, а также еще один общеизвестный факт [7], что $\text{int } l^+ = l^+$ в силу открытости множества l^+, из (14) получаем доказываемое.

4. Покажем, что l^+ – выпукло. Рассмотрим $\forall \overline{x} = (x_1, ..., x_n), \overline{y} = (y_1, ..., y_n) \in l^+$. Тогда, по определению, $l(\overline{x}) = a_1 \cdot x_1 + ... + a_n \cdot x_n + d > 0$, $l(\overline{y}) = a_1 \cdot y_1 + ... + a_n \cdot y_n + d > 0$.

Определим для произвольного $\alpha \in [0; 1]$ величину $\overline{\xi} = (1 - \alpha) \cdot (x_1, ..., x_n) + \alpha \cdot (y_1, ..., y_n) = ((1 - \alpha) \cdot x_1 + \alpha \cdot y_1, ..., (1 - \alpha) \cdot x_n + \alpha \cdot y_n)$.

Тогда из леммы 1 следует, что $l(\overline{\xi}) = (1 - \alpha) \cdot l(\overline{x}) + \alpha \cdot l(\overline{y})$.

Поэтому, если $\alpha \in (0; 1)$, то $l(\overline{\xi}) > 0$, в силу того, что $\alpha > 0, (1-\alpha) > 0$, а также $l(\overline{x}) > 0$ и $l(\overline{y}) > 0$.

Если $\alpha = 0$, то получаем, что $1 - \alpha = 1$ и, следовательно, $l(\overline{\xi}) = (1 - \alpha) \cdot l(\overline{x}) + \alpha \cdot l(\overline{y}) = 1 \cdot l(\overline{x}) + 0 \cdot l(\overline{y}) = l(\overline{x}) > 0$.

Аналогично, если $\alpha = 1$, то $l(\overline{\xi}) = l(\overline{y}) > 0$.

Таким образом, показано, что вместе с любыми двумя точками $\overline{x}, \overline{y} \in l^+$ произвольная точка $\overline{\xi} \in [\overline{x}; \overline{y}]$ также содержится в l^+. Отсюда и следует выпуклость l^+.

Выпуклость множеств $l^-, \overline{l^+}, \overline{l^-}, l$ доказывается аналогично.

□

Лемма 3. *Для всякого разбиения $\{R^1, ..., R^s\}$ пространства $\mathbb{R}^n$ выполнены следующие свойства:*

1) *Любой непустой объемный класс R^i является выпуклым открытым множеством.*

2) *Для любого непустого объемного класса R^i верно, что $\overline{R^i} = R^i \cup \partial_{lin} R^i$.*

3) *Любой непустой плоский класс содержится в замыкании как минимум двух непустых объемных классов эквивалентности.*

Доказательство. 1. Если класс R^i является объемным, то он имеет сигнатуру $\sigma(R^i) = (\sigma_1, ..., \sigma_k)$ такую, что $\forall j \in \{1, ..., k\}$ $\sigma_j \neq 0$. По определению это означает, что класс R^i является решением системы строгих неравенств $\begin{cases} \sigma_1 \cdot l_1(\overline{x}) > 0 \\ ... \\ \sigma_k \cdot l_k(\overline{x}) > 0 \end{cases}$, где $l_j = \{(x_1, ..., x_n) \in \mathbb{R}^n | a_{j1} \cdot x_1 + ... + a_{jn} \cdot x_n + d_j = 0\}, j = 1, ..., k$. Другими словами, R^i является пересечением всех геометрических мест точек $(\sigma_1 \cdot l_1)^+, ..., (\sigma_k \cdot l_k)^+$, где $(\sigma_j \cdot l_j)^+ = \{(x_1, ..., x_n) \in \mathbb{R}^n | \sigma_j \cdot a_{j1} \cdot x_1 + ... + \sigma_j \cdot a_{jn} \cdot x_n + \sigma_j \cdot d_j > 0\}, j = 1, ..., k$.

Далее, так как, по лемме 2, все $(\sigma_j \cdot l_j)^+, j = 1, ..., k$ являются открытыми и выпуклыми множествами, то R^i является открытым и выпуклым множеством, как пересечение конечного числа открытых и выпуклых множеств.

2. Аналогично пункту 1, класс R^i совпадает с множеством решений системы $\begin{cases}\sigma_1 \cdot l_1(\overline{x}) > 0 \\ \ldots \\ \sigma_k \cdot l_k(\overline{x}) > 0\end{cases}$, которое назовем A.

Рассмотрим теперь множество $\overline{A}$, являющееся замыканием множества A, и покажем, что $\overline{A}$ совпадает со множеством всех решений системы $\begin{cases}\sigma_1 \cdot l_1(\overline{x}) \geqslant 0 \\ \ldots \\ \sigma_k \cdot l_k(\overline{x}) \geqslant 0\end{cases}$, которое обозначим B. Доказательство будем вести методом двойного включения.

Докажем включение $\overline{A} \subset B$. Отметим сначала, что верно [8] включение (15).

$$\overline{A} = \overline{(\sigma_1 \cdot l_1)^+ \cap \ldots \cap (\sigma_k \cdot l_k)^+} \subset \overline{(\sigma_1 \cdot l_1)^+} \cap \ldots \cap \overline{(\sigma_k \cdot l_k)^+} \qquad (15)$$

Но по лемме 2 верно, что $\overline{(\sigma_j \cdot l_j)^+} = (\sigma_j \cdot l_j)^+ \cup (\sigma_j \cdot l_j),\ j = 1, \ldots, k$, откуда получаем, что $\overline{(\sigma_j \cdot l_j)^+} = \{\overline{x} \in \mathbb{R}^n | \sigma_j \cdot l_j(\overline{x}) \geqslant 0\},\ j \in 1, \ldots, k$. А отсюда следует, что $\overline{(\sigma_1 \cdot l_1)^+} \cap \ldots \cap \overline{(\sigma_k \cdot l_k)^+} = \{\overline{x} \in \mathbb{R}^n | \sigma_1 \cdot l_1(\overline{x}) \geqslant 0 \& \ldots \& \sigma_k \cdot l_k(\overline{x}) \geqslant 0\}$, то есть $\overline{(\sigma_1 \cdot l_1)^+} \cap \ldots \cap \overline{(\sigma_k \cdot l_k)^+}$ – это множество решений системы $\begin{cases}\sigma_1 \cdot l_1(\overline{x}) \geqslant 0 \\ \ldots \\ \sigma_k \cdot l_k(\overline{x}) \geqslant 0\end{cases}$. Откуда и следует доказываемое включение.

Докажем теперь обратное включение $B \subset \overline{A}$. Для начала возьмем $\forall \overline{x} \in B$. Так как $A \neq \varnothing$, то $\exists \overline{y} \in A$. Причем, так как по пункту 1 данной леммы A – открыто, то вместе с точкой $\overline{y}$ во множестве A содержится целая окрестность $O_\delta(\overline{y})$, поэтому можно выбрать $\overline{y} \in A : \overline{y} \neq \overline{x}$ (см. рис. 2).

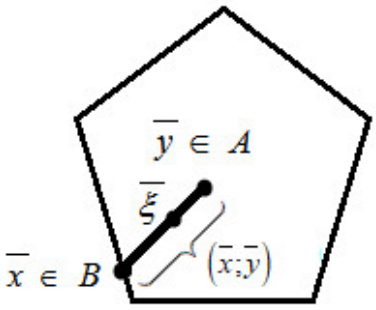


Рис. 2. Построение отрезка, уходящего из $\forall \overline{x} \in B$ внутрь множества A

Рассмотрим множество $N(\overline{x}) = \{j | l_j(\overline{x}) \neq 0\} \subset \{1, ..., k\}$ – множество неактивных ограничений в точке $\overline{x}$. Так как класс R^i – объемный и $\overline{y} \in A = R^i$, то $N(\overline{y}) = \{1, ..., k\}$.

Покажем, что $\forall \overline{\xi} \in (\overline{x}; \overline{y})\ N(\overline{\xi}) = N(\overline{x}) \cup N(\overline{y})$ и $\overline{\xi} \in B$ (см. рис. 2).

Для этого рассмотрим вектор-функцию $r(\overline{x}) = (\sigma_1 \cdot l_1(\overline{x}), ..., \sigma_k \cdot l_k(\overline{x}))$. По построению, для $\forall \overline{x} \in B$ верно, что $r_j(\overline{x}) \geqslant 0, j \in \{1, ..., k\}$. Причем, также для $\forall \overline{x} \in B$, выполняется, что $r_j(\overline{x}) > 0$ тогда и только тогда, когда $l_j(\overline{x}) \neq 0$, то есть $j \in N(\overline{x})$.

Рассмотрим теперь точку $\overline{\xi} \in (\overline{x}; \overline{y})$. По определению это означает, что $\exists t \in (0; 1)\ \overline{\xi} = (1 - t) \cdot \overline{x} + t \cdot \overline{y}$.

Далее, по лемме 1, выполнено равенство (16).

$$r_j(\overline{\xi}) = (1 - t) \cdot r_j(\overline{x}) + t \cdot r_j(\overline{y}), j = 1, ..., k \tag{16}$$

Причем, согласно выбору t, верно, что $t > 0,\ 1 - t > 0$. А так как $\overline{x}, \overline{y} \in B$, то, по определению, $r_j(\overline{x}) \geqslant 0,\ r_j(\overline{y}) \geqslant 0,\ j = 1, ..., k$. Но тогда, в силу (16), выполняется, что $r_j(\overline{\xi}) \geqslant 0, j = 1, ..., k$. Отсюда немедленно следует, что $\overline{\xi} \in B$.

При этом, если хотя бы одна из величин $r_j(\overline{x})$ или $r_j(\overline{y})$ отлична от нуля (учитывая, что данные величины неотрицательны, то сказанное означает, что $r_j(\overline{x}) > 0$ или $r_j(\overline{y}) > 0$), то из (16) следует, что $r_j(\overline{\xi}) > 0$.

Очевидно и обратное, то есть, если $r_j(\overline{\xi}) > 0$, то $r_j(\overline{x}) > 0$ или $r_j(\overline{y}) > 0$.

Из доказанного соотношения $r_j(\overline{\xi}) > 0 \Leftrightarrow r_j(\overline{x}) > 0 \vee r_j(\overline{y}) > 0,\ j = 1, ..., k$ следует, что $\forall \overline{\xi} \in (\overline{x}; \overline{y})\ N(\overline{\xi}) = N(\overline{x}) \cup N(\overline{y})$.

Но тогда, в силу того, что $N(\overline{y}) = \{1, ..., k\}$, получаем, что $N(\overline{\xi}) = \{1, ..., k\}$. Откуда, учитывая дополнительно, что $\overline{\xi} \in B$, получаем, что $\overline{\xi} \in A$.

Теперь рассмотрим $\forall \varepsilon > 0$ и возьмем $t = \frac{\min(\varepsilon, \|\overline{y} - \overline{x}\|)}{2\|\overline{y} - \overline{x}\|} \in (0; 1)$. Ему соответствует некоторая точка $\overline{\xi} = (1 - t) \cdot \overline{x} + t \cdot \overline{y} \in (\overline{x}; \overline{y})$, для которой выполнено, что $\|\overline{\xi} - \overline{x}\| = \|(1 - t) \cdot \overline{x} + t \cdot \overline{y} - \overline{x}\| = \|(1 - t) \cdot \overline{x} + t \cdot \overline{y} - (1 - t) \cdot \overline{x} - t \cdot \overline{x}\| = \|t \cdot (\overline{y} - \overline{x})\| = t \cdot \|\overline{y} - \overline{x}\| \leqslant \frac{\varepsilon}{2\|\overline{y} - \overline{x}\|} \cdot \|\overline{y} - \overline{x}\| = \frac{\varepsilon}{2} < \varepsilon$.

Таким образом, какая бы малая окрестность $O_\varepsilon(\overline{x})$ не была взята, в ней всегда найдется точка $\overline{\xi} \in A$ (см. рис. 3). Отсюда следует, что $\overline{x} \in \overline{A}$.

Итак, показано, что $\overline{A} = B$. Множество B отличается от множества A тем, что в нем добавились классы со всевозможными занулениями сигнатур, то есть классы, обладающие сигнатурами $(\sigma'_1, ..., \sigma'_k) : \sigma'_j \in \{\sigma_j, 0\},\ j = 1, ..., k$. Следовательно, $\overline{A} = A \cup \partial_{lin} A$.

3. Рассмотрим произвольный непустой плоский класс эквивалентности R^i. Без ограничения общности, будем считать, что $\sigma(R^i) = (\sigma_1, ..., \sigma_k)$, где $\sigma_1, ..., \sigma_{k'} \neq 0$, а $\sigma_{k'+1}, ..., \sigma_k = 0$. Причем $k' \in \{1, ..., k - 1\}$, так как в противном случае взятый класс является объемным. Тогда,

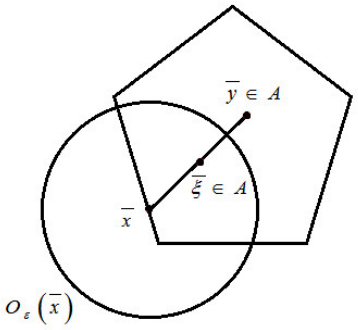


Рис. 3. Иллюстрация того, что $\forall O_\varepsilon(\overline{x}) \exists \overline{\xi} \in O_\varepsilon(\overline{x}) \cap (\overline{x}; \overline{y})$

по определению, R^i является решением системы равенств и неравенств (17), которую можно переписать в виде (18).

$$\begin{cases} \sigma_1 \cdot l_1(\overline{x}) > 0 \\ \dots \\ \sigma_{k'} \cdot l_{k'}(\overline{x}) > 0 \\ \sigma_{k'+1} \cdot l_{k'+1}(\overline{x}) = 0 \\ \dots \\ \sigma_k \cdot l_k(\overline{x}) = 0 \end{cases} \tag{17}$$

$$\begin{cases} \sigma_1 \cdot l_1(\overline{x}) > 0 \\ \dots \\ \sigma_{k'} \cdot l_{k'}(\overline{x}) > 0 \\ \sigma_{k'+1} \cdot l_{k'+1}(\overline{x}) \geqslant 0 \\ \dots \\ \sigma_k \cdot l_k(\overline{x}) \geqslant 0 \\ \sigma_{k'+1} \cdot l_{k'+1}(\overline{x}) \leqslant 0 \\ \dots \\ \sigma_k \cdot l_k(\overline{x}) \leqslant 0 \end{cases} \tag{18}$$

Рассмотрим теперь систему (19), полученную из системы (18) заменой всех строгих неравенств на нестрогие.

$$\begin{cases} \sigma_1 \cdot l_1(\overline{x}) \geqslant 0 \\ ... \\ \sigma_{k'} \cdot l_{k'}(\overline{x}) \geqslant 0 \\ \sigma_{k'+1} \cdot l_{k'+1}(\overline{x}) \geqslant 0 \\ ... \\ \sigma_k \cdot l_k(\overline{x}) \geqslant 0 \\ \sigma_{k'+1} \cdot l_{k'+1}(\overline{x}) \leqslant 0 \\ ... \\ \sigma_k \cdot l_k(\overline{x}) \leqslant 0 \end{cases} \tag{19}$$

Очевидно, что множество решений системы (19) содержится в пересечении замыканий объемных классов R^{j_1}, R^{j_2}, задаваемых системами (20) и (21), соответственно.

$$\begin{cases} \sigma_1 \cdot l_1(\overline{x}) > 0 \\ ... \\ \sigma_{k'} \cdot l_{k'}(\overline{x}) > 0 \\ \sigma_{k'+1} \cdot l_{k'+1}(\overline{x}) > 0 \\ ... \\ \sigma_k \cdot l_k(\overline{x}) > 0 \end{cases} \tag{20}$$

$$\begin{cases} \sigma_1 \cdot l_1(\overline{x}) > 0 \\ ... \\ \sigma_{k'} \cdot l_{k'}(\overline{x}) > 0 \\ \sigma_{k'+1} \cdot l_{k'+1}(\overline{x}) < 0 \\ ... \\ \sigma_k \cdot l_k(\overline{x}) < 0 \end{cases} \tag{21}$$

Таким образом, получаем, что класс R^i, являющийся множеством решений системы (18), содержится во множестве решений системы (19), которое, как было сказано выше, содержится в пересечении $\overline{R^{j_1}} \cap \overline{R^{j_2}}$. Отсюда следует, что $R^i \subset \overline{R^{j_1}} \cap \overline{R^{j_2}}$. Причем, если хотя бы один из классов R^{j_1} или R^{j_2} оказался пуст, то тогда и $\overline{R^{j_1}} \cap \overline{R^{j_2}}$ было бы пусто, а, следовательно, R^i также был бы пустым, что противоречит условию. □

Лемма 4. *Пусть $f : \mathbb{R}^n \to \mathbb{R}$ – выпуклая непрерывная кусочно-линейная функция, заданная над разбиением $\{R^1, ..., R^s\}$ пространства $\mathbb{R}^n$, а R^i, R^j – два произвольных непустых объемных класса из данного разбиения.*

Тогда верно, что $\forall \overline{x} \in R^j \ f(\overline{x}) \geqslant f_{R^i}(\overline{x})$ *(см. рис. 4).*

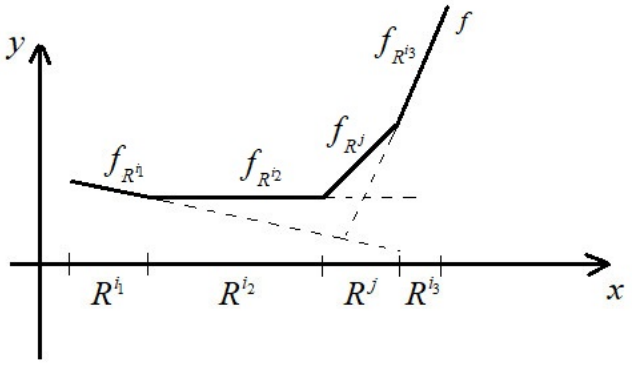


Рис. 4. Пример-иллюстрация утверждения леммы (одномерный случай). В качестве R^i можно взять, например, $R^{i_1}, R^{i_2}, R^{i_3}$

Доказательство. Предположим противное, что нашлись такие непустые объемные классы R^i, R^j, что верно выражение (22).

$$\exists \overline{x} \in R^j \ f(\overline{x}) < f_{R^i}(\overline{x}) \tag{22}$$

Очевидно, что $i \neq j$, так как в противном случае выполняется, что $\forall \overline{x} \in R^j \ f(\overline{x}) = f_{R^i}(\overline{x})$, что противоречит предположению (22).

Так как по условию $\forall \overline{x} \in R^j \ f(\overline{x}) = f_{R^j}(\overline{x})$, то неравенство (22) можно записать, в виде (23).

$$\exists \overline{x} \in R^j \ f_{R^j}(\overline{x}) < f_{R^i}(\overline{x}) \tag{23}$$

Зафиксируем $\overline{x} \in R^j$, для которого выполняется предположение (22) и $\forall \overline{y} \in R^i$, после чего соединим их отрезком $[\overline{x}; \overline{y}]$.

Обозначим $f(t) = f((1-t) \cdot \overline{x} + t \cdot \overline{y})$, $f_{R^i}(t) = f_{R^i}((1-t) \cdot \overline{x} + t \cdot \overline{y})$ и $f_{R^j}(t) = f_{R^j}((1-t) \cdot \overline{x} + t \cdot \overline{y})$, где $t \in \mathbb{R}$, сечения функций $f(\overline{x}), f_{R^i}(\overline{x}), f_{R^j}(\overline{x})$ прямыми, проходящими через зафиксированные точки $\overline{x}, \overline{y}$.

Отметим, что $f(t)$ – выпуклая, так как она является сужением выпуклой функции $f(\overline{x})$ на выпуклое множество, а функции $f_{R^i}(t)$ и $f_{R^j}(t)$ являются линейными функциями одного аргумента $t \in \mathbb{R}$.

Рассмотрим функцию $A(t) = f_{R^i}(t) - f_{R^j}(t)$, которая, очевидно, также линейна, причем в силу (23) выполняется (24).

$$A(0) > 0 \tag{24}$$

Далее рассмотрим два случая:

• Пусть $\forall t \in [0;1)\ A(t) \neq 0$. Но тогда, в силу (24) и непрерывности функции $A(t)$ получаем, что $\forall t \in [0;1)\ A(t) > 0$.

По определению, $\forall \overline{x} \in R^j\ f(\overline{x}) = f_{R^j}(\overline{x})$, $\forall \overline{y} \in R^i\ f(\overline{y}) = f_{R^i}(\overline{y})$. При этом R^i, R^j – выпуклые, открытые и $R^j \cap R^i = \varnothing$. Поэтому $\exists t_1, t_2 \in (0;1)$: $t_2 > t_1$ и при этом $\forall t \in [0;t_1]\ f(t) = f_{R^j}(t)$, а также $\forall t \in [t_2;1]\ f(t) = f_{R^i}(t)$ (см. рис. 5).

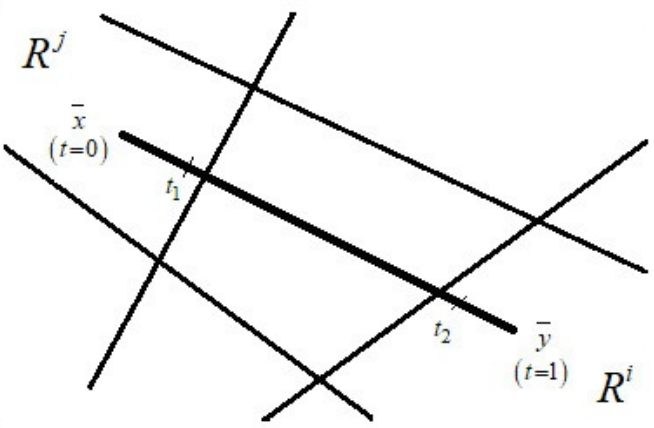


Рис. 5. Иллюстрация к утверждению о том, что $\forall t \in [0;t_1]\ f(t) = f_{R^j}(t)$ и $\forall t \in [t_2;1]\ f(t) = f_{R^i}(t)$

Рассмотрим отрезок $[t_1;1] \subset \mathbb{R}$ и выразим t_2 в виде следующей комбинации границ данного отрезка: $t_2 = (1-\alpha) \cdot t_1 + \alpha \cdot 1$, где, очевидно, $\alpha \in (0;1)$. Отметим, что по лемме 1 верно выражение (25).

$$\forall \alpha \in [0;1]\ (1-\alpha) \cdot f_{R^i}(t_1) + \alpha \cdot f_{R^i}(1) = f_{R^i}((1-\alpha) \cdot t_1 + \alpha \cdot 1) \qquad (25)$$

Учитывая (25) и тот факт, что $f_{R^i}(t_1) > f_{R^j}(t_1)$, который следует из предположения $\forall t \in [0;1)\ A(t) > 0$, сделанного в начале доказательства этого случая, получаем, что верна следующая цепочка преобразований:

$$\begin{aligned}(1-\alpha) \cdot f(t_1) + \alpha \cdot f(1) &\overset{def}{=} (1-\alpha) \cdot f_{R^j}(t_1) + \alpha \cdot f_{R^i}(1) < \\ &< (1-\alpha) \cdot f_{R^i}(t_1) + \alpha \cdot f_{R^i}(1) = f_{R^i}((1-\alpha) \cdot t_1 + \alpha \cdot 1) = \\ &= f_{R^i}(t_2) = f(t_2) = f((1-\alpha) \cdot t_1 + \alpha \cdot 1) \qquad (26)\end{aligned}$$

Полученное неравенство (26) противоречит выпуклости функции $f(t)$.

• Теперь рассмотрим случай, когда $\exists t^* \in [0;1) :\ A(t^*) = 0$. Отметим, что $A(0) > 0$. Отсюда и из предыдущего немедленно следует, что данная линейная функция одного аргумента имеет ненулевой угловой коэффициент и является строго монотонной. Поэтому $\forall t : t > t^*$ выполняется, что $A(t) < 0$. В частности, $A(1) < 0$, откуда следует, что $f_{R^i}(1) < f_{R^j}(1)$.

Но тогда для $\forall t \in (0;1)$ выполняется, что $(1-t) \cdot f(0) + t \cdot f(1) = (1-t) \cdot f_{R^j}(0) + t \cdot f_{R^i}(1) < (1-t) \cdot f_{R^j}(0) + t \cdot f_{R^j}(1) = f_{R^j}((1-t) \cdot 0 + t \cdot 1) = f((1-t) \cdot 0 + t \cdot 1)$.

Таким образом, снова имеем противоречие с выпуклостью функции $f(t)$.

Учитывая, что всегда выполняется один из рассмотренных случаев, получаем доказываемое. □

Теорема 1 (о выпуклых CPL-функциях)**.** *Любую выпуклую непрерывную кусочно-линейную функцию $f : \mathbb{R}^n \to \mathbb{R}$, заданную над разбиением $\{R^1, ..., R^s\}$, можно выразить нейронной схемой нелинейной сложности 1 и нелинейной глубины 1 над базисом B_1.*

Доказательство. Отметим сначала, что при задании функции f в действительности используются лишь непустые классы эквивалентности. Это происходит в силу того, что определения вида $\forall \overline{x} \in R^j\ f(\overline{x}) = f_{R^j}(\overline{x})$ попросту не имеют смысла, если класс эквивалентности R^j – пуст.

Таким образом, если показать равенство функции f и некоторой функции g на точках всех непустых классов эквивалентности, то сразу же будет показано равенство данных функций.

Рассмотрим функцию $\max\limits_{j \in \{1,...,s'\}} f_{R^j}(\overline{x})$, где $R^1, ..., R^{s'}$ – все непустые объемные классы эквивалентности разбиения.

Далее возьмем произвольный непустой класс эквивалентности R^j. Возможны два случая:

• R^j – объемный. Тогда, так как функция $f(\overline{x})$ выпукла, то из леммы 4 следует, что $\forall \overline{x} \in R^j\ f(\overline{x}) \geqslant f_{R^{j'}}(\overline{x}), j' \in \{1, ..., s'\}$.

При этом $\exists j' = j : \forall \overline{x} \in R^j\ f(\overline{x}) = f_{R^j}(\overline{x})$, то есть среди объемных классов $\{R^1, ..., R^{s'}\}$ существует класс R^j, на котором достигается равенство функции $f(\overline{x})$ и функции $f_{R^j}(\overline{x})$, соответствующей данному классу. Отсюда следует, что $f(\overline{x}) = \max\limits_{j \in \{1,...,s'\}} f_{R^j}(\overline{x})$.

• R^j – плоский. По лемме 3 данный класс содержится в замыкании некоторого непустого объемного класса эквивалентности R^i. Последнее означает, что $\forall \overline{x} \in R^j\ \exists \{\overline{x_n}\}_{n \in \mathbb{N}} \subset R^i : \overline{x} = \lim\limits_{n \to +\infty} \overline{x_n}$.

По ранее доказанному, на любом непустом объемном классе эквивалентности выполняется, что $f(\overline{x}) = \max\limits_{j\in\{1,\dots,s'\}} f_{R^j}(\overline{x})$, откуда следует выражение (27).

$$\forall n \in \mathbb{N}\ f(\overline{x_n}) = \max_{j\in\{1,\dots,s'\}} f_{R^j}(\overline{x_n}) \tag{27}$$

Переходя в выражении (27) к пределу при $n \to +\infty$, получаем выражение (28).

$$\lim_{n\to+\infty} f(\overline{x_n}) = \lim_{n\to+\infty} \max_{j\in\{1,\dots,s'\}} f_{R^j}(\overline{x_n}) \tag{28}$$

В выражении (28), в силу непрерывности функций $f(\overline{x})$, $\max\limits_{j\in\{1,\dots,s'\}} f_{R^j}(\overline{x_n})$, можно перейти от пределов функций к пределу аргументов данных функций. В результате данного перехода получаем выражение (29).

$$f(\lim_{n\to+\infty} \overline{x_n}) = \max_{j\in\{1,\dots,s'\}} f_{R^j}(\lim_{n\to+\infty} \overline{x_n}) \tag{29}$$

Но $\lim\limits_{n\to+\infty} \overline{x_n} = \overline{x}$, поэтому, переходя к пределу в выражении (29), получаем выражение (30).

$$f(\overline{x}) = \max_{j\in\{1,\dots,s'\}} f_{R^j}(\overline{x}) \tag{30}$$

Таким образом, показано, что для любого непустого класса эквивалентности выполняется выражение (30). Откуда и следует доказываемое утверждение. □

Следствие 1. *Если в условиях теоремы 1 заменить базис B_1 на B_2, то в новом базисе любую выпуклую непрерывную кусочно-линейную функцию $f : \mathbb{R}^n \to \mathbb{R}$, заданную над разбиением пространства $\mathbb{R}^n$, в котором содержится s' непустых объемных классов эквивалентности, можно выразить нейронной схемой нелинейной глубины $\lceil \log_2 s' \rceil$ и нелинейной сложности $s' - 1$.*

Причем полученные оценки нелинейной сложности и глубины являются неулучшаемыми.

Доказательство. Отметим сначала, что в базисе B_2 можно выразить функцию $\max(x, y)$. Нетрудно убедиться, что данная функция выражается по формуле (31).

$$\max(x, y) = RELU(x - y) + y \tag{31}$$

Но имея функцию $\max(x, y)$, можно оптимальным образом [9] выразить функцию $\max(x_1, ..., x_{s'})$, объединяя аргументы по сбалансированному бинарному дереву [10], [11], высота которого, как известно [11], равна $\lceil \log_2 s' \rceil$ и количество элементов $\max(x, y)$ в котором равно $s' - 1$.

Далее, заменяя $\max(x, y)$ по формуле (31), получаем нейронную схему той же нелинейной сложности и глубины (так как вместо каждого максимума при замене возникает одна функция $RELU$ и какие-то линейные функции, то есть число нелинейных элементов при данном преобразовании не меняется).

При этом, очевидно, что при замене аргументов функции $\max(x_1, ..., x_{s'})$ на соответствующие им значения линейных функций $f_{R^j}(\overline{x})$ в формуле (30) новые нелинейные элементы в конструируемую нейронную схему не добавляются.

Таким образом, заданную в условии кусочно-линейную функцию, можно представить в виде нейронной схемы нелинейной глубины $\lceil \log_2 s' \rceil$ и нелинейной сложности $s' - 1$. □

Отметим, что существуют и вогнутые (выпуклые вверх) функции. Пусть дана вогнутая функция $f(\overline{x})$ класса CPL. Тогда $-f(\overline{x})$ является выпуклой функцией класса CPL. Но из теоремы 1 и следствия 1 следует, что всякая выпуклая CPL-функция представима в виде некоторой нейронной схемы $g_1(\overline{x})$ нелинейной сложности 1 над базисом B_1 и некоторой нейронной схемы $g_2(\overline{x})$ над базисом B_2 нелинейной сложности $s' - 1$ и нелинейной глубины $\lceil \log_2 s' \rceil$.

Но тогда $-1 \cdot g_1(\overline{x})$ и $-1 \cdot g_2(\overline{x})$ являются нейронными схемами той же нелинейной сложности и глубины, что и схемы $g_1(\overline{x})$, $g_2(\overline{x})$, но при этом данные схемы задают уже функцию $f(\overline{x})$.

Из вышесказанного вытекает, что все доказанные утверждения верны и для любой вогнутой CPL-функции.

Отметим также, что класс CPL-функций состоит не только из выпуклых или вогнутых CPL-функций. Так, например, функция $f(x) = x - 2 \cdot RELU(x) + 2 \cdot RELU(x - 1)$, очевидно, является CPL-функцией, но не является ни выпуклой, ни вогнутой функцией (см. рис. 6).

Автор выражает благодарность научному руководителю, к.ф.-м.н., научному сотруднику В.С. Половникову за постановку задачи и научное руководство, а также д.ф.-м.н., доценту А.А. Часовских за помощь в проверке полученных результатов.

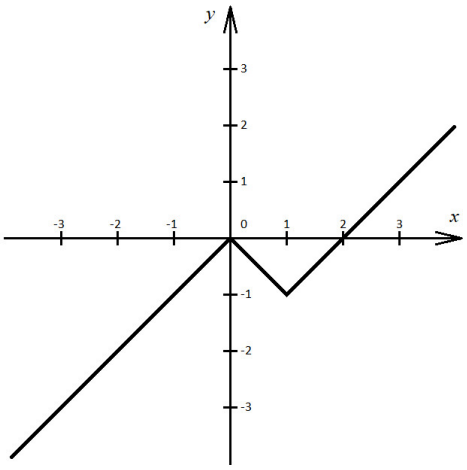


Рис. 6. CPL-функция, не являющаяся выпуклой или вогнутой

Список литературы

[1] Половников В.С., *Об оптимизации структурной реализации нейронных сетей*, Диссертация на соискание степени кандидата наук, Москва, 2007, 96 с.

[2] Яблонский С.В., *Введение в дискретную математику*, Наука. Гл. ред. физ.-мат. лит., Москва, 1986, 384 с.

[3] Simon Haykin, *Neural Networks. A Comprehensive Foundation*, Prentice Hall International, Inc., Canada, 1999, 1999 с.

[4] Шишляков В.Г., "О построении явной архитектуры нейронной сети, приближающей кусочно-линейные функции", *Интеллектуальные системы. Теория и приложения*, **26**:2 (2022), 42–60.

[5] McCulloch W.S., Pitts W., "A logical calculus of the ideas immanent nervous activity", *Bull. Of math. Biophysics*, **5** (1943), 115–133.

[6] Кан А.Н., "Вопросы полноты в классе кусочно-линейных непрерывных функций", *Интеллектуальные системы. Теория и приложения*, **21**:2 (2017), 46–56.

[7] Мищенко А.С., Фоменко А.Т., *Курс дифференциальной геометрии и топологии*, Факториал Пресс, Москва, 2000, 448 с.

[8] Виро О.Я., Иванов О.А., Нецветаев Н.Ю., Харламов В.М., *Задачи по топологии*, СПбГУ, Санкт-Петербург, 2000, 208 с.

[9] Абельхарисов А.В., “О существовании нейронных схем произвольной минимальной нелинейной глубины в базисе с функцией активации ReLU”, *Ломоносовские чтения 2021*, 2021.

[10] Кнут Д., *Искусство программирования. Том 1. Основные алгоритмы*, Вильямс, Москва, 2019, 720 с.

[11] Седжвик Р., *Фундаментальные алгоритмы на C++*, ДиаСофт, Санкт-Петербург, 2001, 688 с.

Convex CPL-functions recovering by neural networks on RELU-bases
Shishlyakov V.G.

The present paper considers a problem of functional classes obtained by using neural networks on max non-linearities bases. Firstly, some properties of CPL-functions and equivalence classes generating them are investigated. Proceeding from these properties a theorem is proved that neural networks built on the basis of linear and max non-linearity functions can exactly recover any convex CPL-function.

Secondly, RELU-basis, a special case of max non-linearities bases, is investigated, with a theorem similar to the previous one mentioned above proved. The question of estimating the number of neurons and layers in obtained architectures is also discussed.

All the mentioned theorems have a constructive proof, i.e. neural network architectures with mentioned features are built explicitly.

Keywords: Neural networks, architecture, functions recovery, functions expressibility, convex functions, particle-linear functions, ReLU function, max function.

References

[1] Polovnikov V.S., *On optimization of the structural implementation of neural networks*, Ph.D. Thesis ... physical and mathematical sciences, MSU, Moscow, 2007 (In Russian)

[2] YAblonskij S.V., *Introduction to discrete mathematics*, eds. fiz.-mat.lit., «Science», Moscow, 1986 (In Russian)

[3] Simon Haykin, *Neural Networks. A Comprehensive Foundation*, Prentice Hall International, Inc., Canada, 1999, 1999 pp.

[4] Shishlyakov V.G., “On the construction of an explicit neural network architecture that approximates particle-linear functions”, *Intelligent systems. Theory and applications.*, **26**:2 (2022), 42–60 (In Russian)

[5] McCulloch W.S., Pitts W., “A logical calculus of the ideas immanent nervous activity”, *Bull. Of math. Biophysics*, **5** (1943), 115–133

[6] Kan A.N., “Questions of completeness in the class of particle-linear continuous functions”, *Intelligent systems. Theory and applications.*, **21**:2 (2017), 46–56 (In Russian)

[7] Mishchenko A.S., Fomenko A.T., *Course in Differential Geometry and Topology*, Factorial Press, Moscow, 2000 (In Russian)

[8] Viro O.Ya., Ivanov O.A., Netsvetaev N.Yu., Kharlamov V.M., *Topology tasks*, St. Petersburg State University, St. Petersburg, 2000 (In Russian)

[9] Abelkharisov A.V., “On the Existence of Neural Circuits of Arbitrary Minimum Nonlinear Depth in a Basis with ReLU Activation Function”, *Lomonosov Readings 2021*, 2021 (In Russian)

[10] Knut D., *The art of programming. Volume 1. Basic Algorithms*, Williams, Moscow, 2019 (In Russian)

[11] Sedgwick R., *Fundamental Algorithms in C++*, DiaSoft, St. Petersburg, 2001 (In Russian)

К сведению авторов публикаций в журнале «Интеллектуальные системы. Теория и приложения»

В соответствии с требованиями ВАК РФ к изданиям, входящим в перечень ведущих рецензируемых научных журналов и изданий, в которых могут быть опубликованы основные научные результаты диссертаций на соискание ученой степени доктора и кандидата наук, статьи в журнал «Интеллектуальные системы. Теория и приложения» предоставляются авторами в следующей форме:

1. Статьи, набранные в пакете LaTeX, предоставляются к загрузке через WEB-форму http://intsysjournal.org/generator_form .
2. К статье прилагаются файлы, содержащие название статьи на русском и английском языках, аннотацию на русском и английском языках (не более 50 слов), список ключевых слов на русском и английском языках (не более 20 слов), информация об авторах: Ф.И.О. полностью, место работы, должность, ученая степень и/или звание (если имеется), контактные телефоны (с кодом города и страны), e-mail, почтовый адрес с индексом города (домашний или служебный).
3. Список литературы оформляется в едином формате, установленном системой Российского индекса научного цитирования.
4. За публикацию статей в журнале «Интеллектуальные системы. Теория и приложения» с авторов (в том числе аспирантов высших учебных заведений) статей, рекомендованных к публикации, плата не взимается. Оттиски статей авторам не предоставляются. Журнал распространяется по подписке, экземпляры журнала рассылаются подписчикам наложенным платежом. Условия подписки публикуются в каталоге НТИ «Роспечать», индекс журнала 64559.
5. Доступ к электронной версии последнего вышедшего номера осуществляется через НЭБ «Российский индекс научного цитирования». Номера, вышедшие ранее, размещаются на сайте http://intsysjournal.org, и доступ к ним бесплатный. Там же будут размещены аннотации всех публикуемых статей.

Подписано в печать: 20.12.2022
Дата выхода: 25.12.2022
Тираж: 200 зкз.
Цена свободная
Свидетельство о регистрации СМИ: ПИ № ФС77–58444 от 25 июня 2014 г., выдано Федеральной службой по надзору в сфере связи, информационных технологий и массовых коммуникаций(Роскомнадзор).